KARNAVATI UNIVERSITY

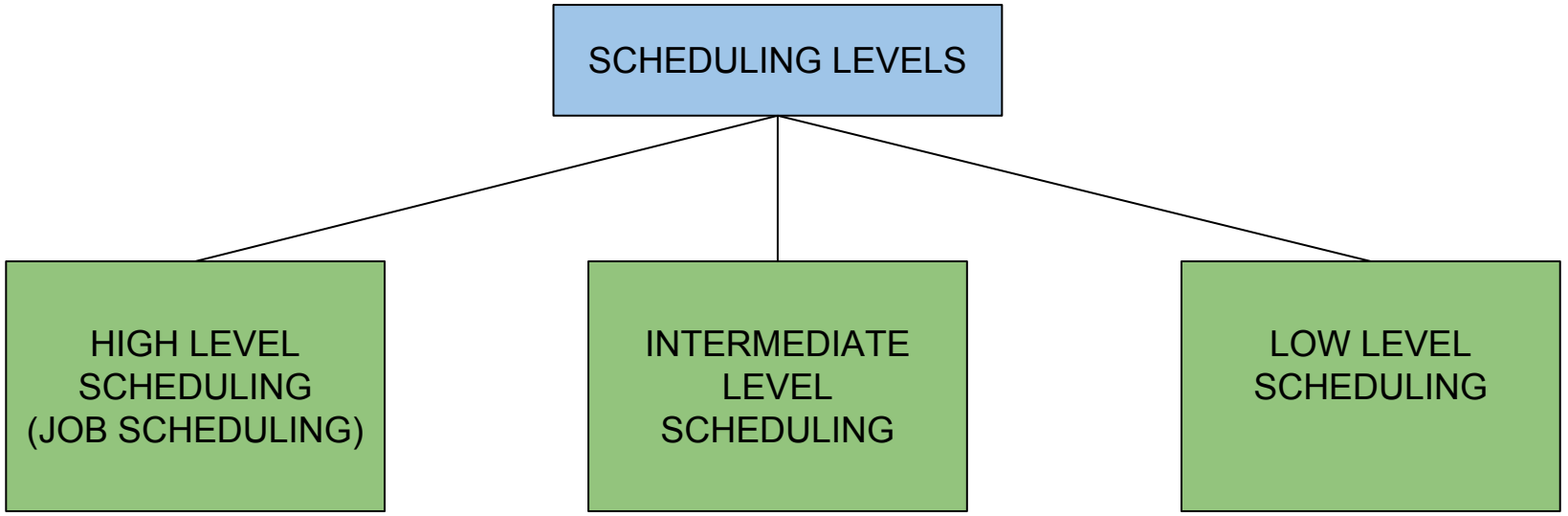


# Operating Systems
Course Code: **71203002004**
**Scheduling**

***by -***
***Minal Rajwar***

# Scheduling

- When a system has choices to execute, it must have a strategy - called a processor scheduling policy(process).
- In order to run smoothly it has to satisfy a certain performance criteria, such that maximum number of processes can complete per unit time.



# High Level Scheduling

- also called job scheduling or long term scheduling.
- determines which job system allows for efficient use of system resources.
- Sometimes called admission scheduling system - determine which job gain access to system.
- dictates level of multiprogramming - total number of process in the system at a given time.
- entry of too many process at a time saturate system resources leading to poor performance.

High level scheduling policy may decide to temporarily prohibit new jobs from entering until other jobs complete.

# Intermediate Level Scheduling

- after the job has been admitted
- determines which processes shall be allowed to complete for processors.
- responds to short term fluctuations in system load.
- temporarily suspends and resumes processes to perform smooth operation and realize performance goals.

It is a buffer between admissions of jobs to the system and the assignment of processors to the processes representing these jobs.

# Low Level Scheduling

- also called the dispatcher.
- determines which active process the system will assign to a processor when the next becomes available.
- assign a priority to each process, reflecting its importance - more important process gets executed first.

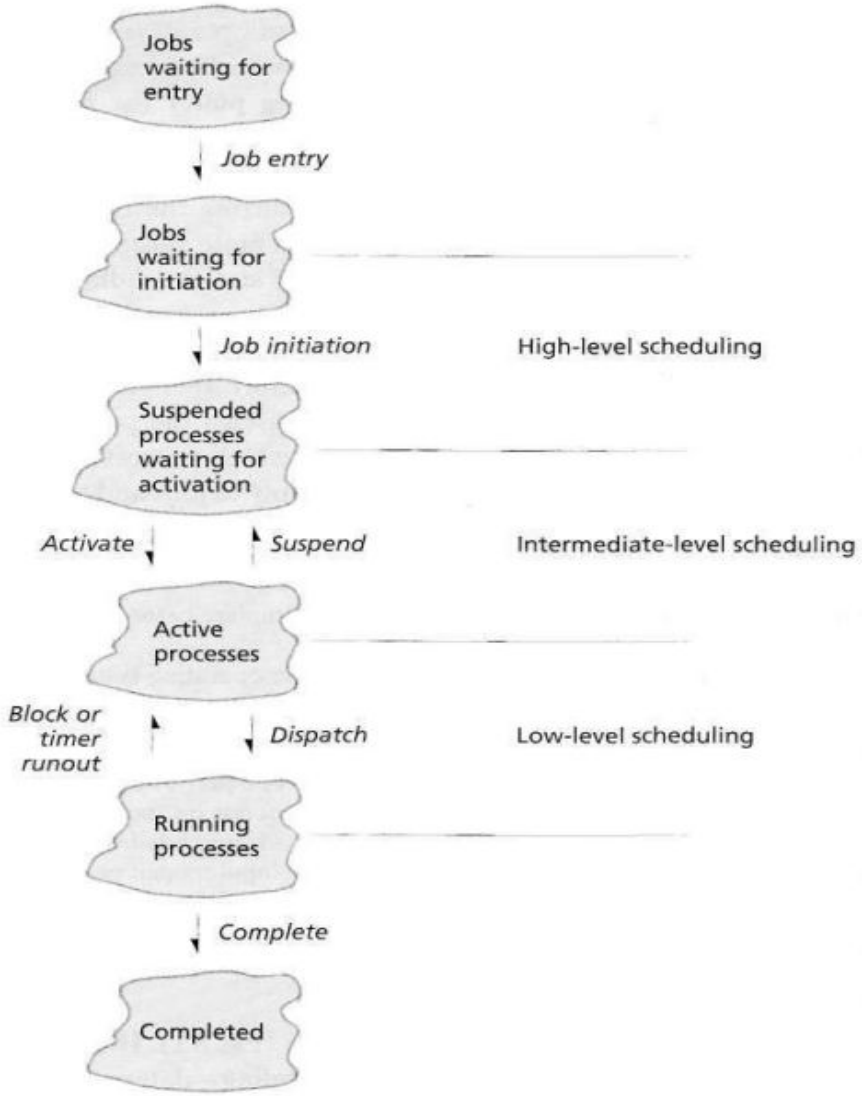
operates many times per second

resides in main memory.

High Level -
select jobs allowed to compete for CPU and other system resources.

Intermediate Level -
select which jobs to temporarily suspend/ resume to smooth fluctuations in system load.

Low Level -
selects the ready process that will be assigned the CPU resources.



# Preemptive vs Non-Preemptive Scheduling

**Non Preemptive**

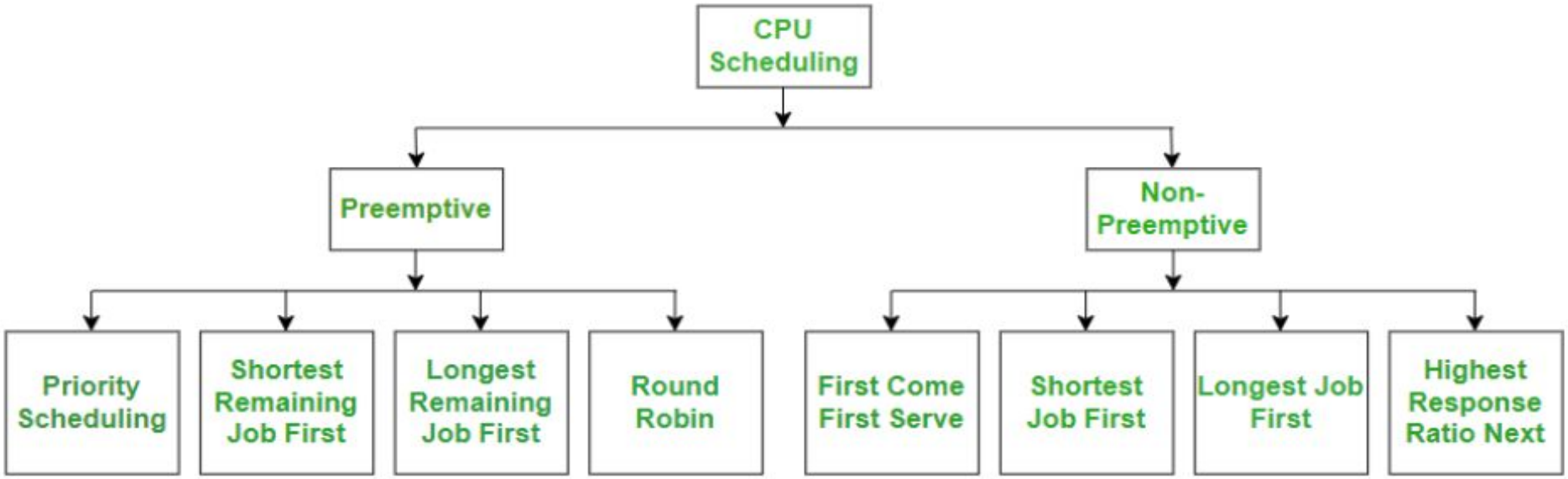
Once the system has assigned a processor to a process, it cannot be removed or taken back.

Process runs till executions or until it willingly give up its resources.

**Preemptive**

System can remove the processor from the process it is running.

Is useful where high priority processes require rapid response. (may cause overhead)


CPU Scheduling
Preemptive
Non-Preemptive
Priority Scheduling
Shortest Remaining Job First
Longest Remaining Job First
Round Robin
First Come First Serve
Shortest Job First
Longest Job First
Highest Response Ratio Next

# Preemptive Scheduling Example

Imagine you're using your smartphone. You're watching a YouTube video (Process A), and suddenly a phone call comes in (Process B).

- **Process A** (YouTube) is running.
- **Process B** (incoming call) has **higher priority**.
- The system **pauses** the YouTube video and **switches** to the call screen.

The operating system **preempts** the running task (video) to give control to the higher-priority task (call). Once the call ends, the video resumes.

# Non Preemptive Scheduling Example

You send a large document (Job A) to a shared printer. A colleague sends a short one-page document (Job B) right after you.

- Job A starts printing.
- Even though Job B is shorter, it has to **wait** until Job A finishes.

The printer doesn't interrupt a job once it starts — it runs **to completion**. This is classic **non-preemptive behavior**.

# Priorities

Schedulers use priorities to manage process execution.

Static -
fixed, low overhead

Dynamix -
adjustable, more responsive but complex.

Dynamic Priorities adapt to system changes like boosting a process's priority when it holds a resource needed by a higher-priority task, then lowering it afterward.


In multiuser systems, the OS must balance fairness and responsiveness.

Users may pay for higher priority, justifying the reallocation of resources.

Without a cost, everyone would demand top service, leading to misuse.

# Scheduling Objectives

When designing a scheduling discipline for a system, several factors must be considered, including system type (eg., real-time vs desktop) and user expectations.

The scheduler may aim to

- Maximize Throughput: complete as many processes as possible per time unit.
- Avoid Indefinite Postponement - ensure all processes eventually get CPU time.
- Improve response time - especially for interactive processes.
- Enforce Priorities - higher-priority tasks should be served first.
- Maximize resource use - keep CPU and resource active
- Minimize Overhead - While not a main goal, lower overhead improve efficiency.
- Ensure Predictability - keep process execution times consistent.

Techniques Like

Aging
(gradually raising priority of waiting tasks)

Priority Inversion
(temporarily boosting lower priority tasks to free key resources)

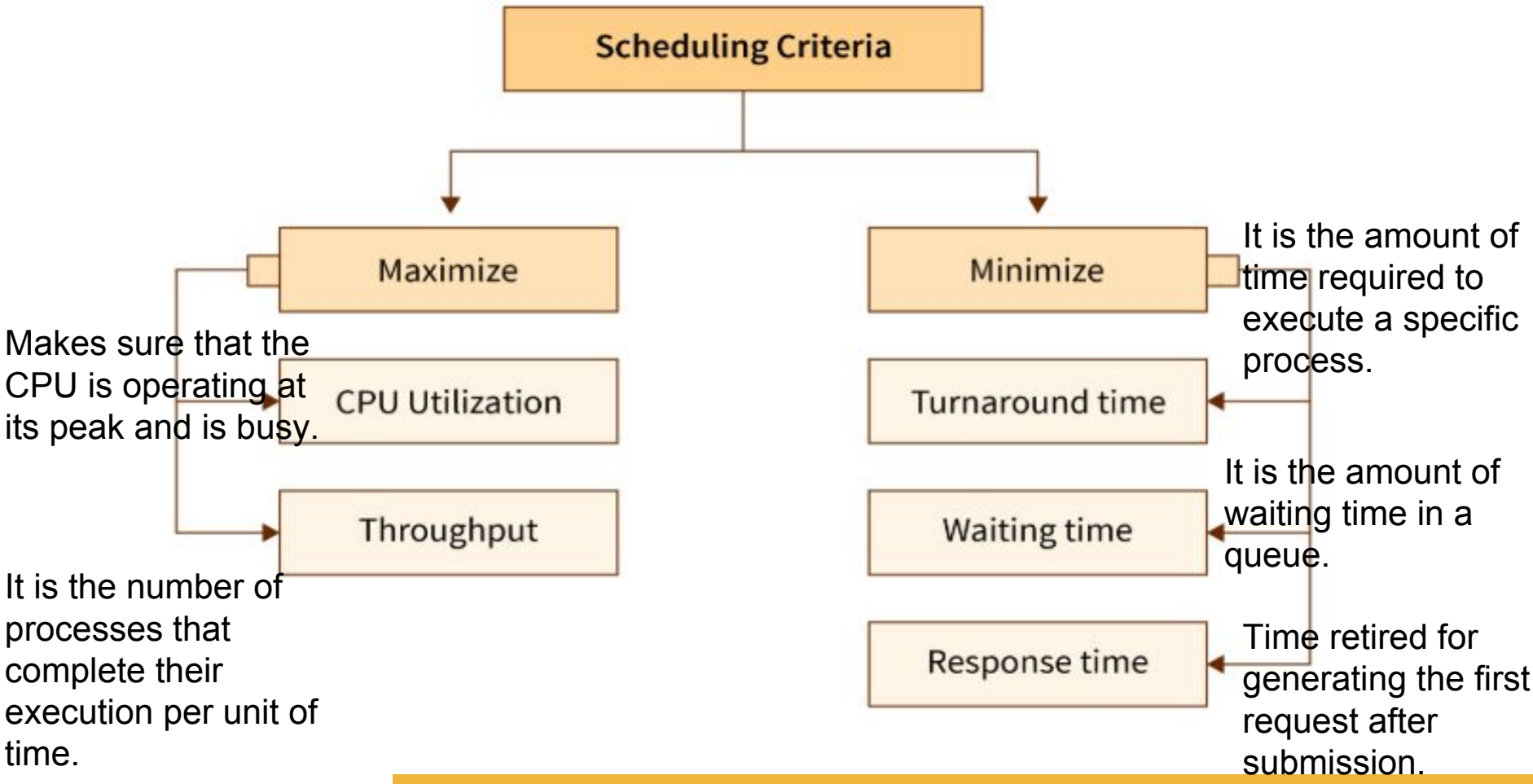
help meet these goals.

Goals often conflict - e.g., fast response vs. efficient resource use- making scheduling complex.

Common desirable properties of good scheduling disciplines include:

- Fairness: Equal treatment for similar processes.
- Predictability: Consistent performance.
- Scalability: Stable behavior under heavy load.

# Scheduling Criteria



# DISCUSSION & REVISION

1. Which level of scheduler is also called a dispatcher?
2. What are two types of CPU scheduling techniques?
3. What is throughput?
4. What is the aim of the scheduler?
5. Which CPU scheduling level decides which process will be assigned with the CPU resources?