Threading & Execution Notes

1. What is a Thread? How is it Different from a Process?

- Thread: The smallest unit of execution within a process.

- Process: An independent program in execution with its own memory space.

Feature	Process	Thread
Memory	Has its own memory space	Shares memory with others
Overhead	Heavy	Lightweight
Communication	Needs IPC (pipes, sockets)	Shared memory (easier)
Isolation	Isolated from others	Not isolated

2. What is Multithreading?

- Multithreading is the ability of a CPU or process to manage multiple threads simultaneously.

- Used for better resource utilization and parallelism.

3. Thread States (Lifecycle):

1. New - Thread is created but not started
2. Runnable - Ready to run, waiting for CPU time
3. Running - Actively executing
4. Blocked/Waiting - Waiting for a monitor lock or I/O
5. Timed Waiting - Waiting for a specified time
6. Terminated - Execution finished

4. What is Hyper-Threading?

- Intel's implementation of Simultaneous Multithreading (SMT).

- Allows one physical core to behave like two logical cores.

- Boosts performance by utilizing idle execution units of a core.

- Improves throughput, not true doubling of speed.

5. CPU-bound vs I/O-bound Threads

Type	Description	Examples
CPU-bound	Heavy CPU usage	Video encoding, sorting

I/O-bound	Waiting on external systems	File I/O, DB, HTTP calls

- CPU-bound tasks need CPU cores to perform computation.

- I/O-bound tasks yield CPU when waiting, freeing resources.

6. Thread Scheduling

- Managed by the Operating System (OS).
- Determines which thread gets CPU time using strategies like:
  - Round Robin
  - Priority Scheduling
  - Fair Share

7. Context Switching

- Saving the state of a running thread and restoring another's state.
- Enables multiple threads to share a CPU core.
- Has overhead due to register saving/loading and memory context.

8. How Thousands of Threads Run in Parallel

- OS uses context switching and scheduling to rotate threads.
- Even on limited cores, threads get small time slices (milliseconds).
- I/O-bound threads free CPU while waiting.
- This makes it seem like all threads run in parallel.