Context Switching

What is Context Switching?

Context Switching means the CPU **pauses one thread**, saves its current state, and **switches** to another thread.

Why too many threads = too much context switching?

Imagine you create **200 threads** for 200 tasks.
But your CPU has only **8 cores**.
So the CPU must continuously:

- Pause thread A
- Save its state
- Load thread B
- Run it
- Pause thread B
- Load thread C
  … and so on

This rapid switching happens **hundreds of times per second**, causing slow performance.
Too many threads → too many switches → CPU waste → slow program.

Think of it like:

You're chatting with two people on WhatsApp.

When you switch from Person A to Person B:

- You remember where you stopped (context).
- You continue the other chat.
- Later you come back and resume.

That's exactly what the CPU does with threads.

What exactly is saved in "Context"?

When a thread is paused, the OS stores:

- Program counter (which line to execute next)
- Register values
- Stack pointer
- Thread state

Then it loads the state of another thread and continues.

Why does Context Switching happens ?

The Reason why Context switching happens because:

—> Multiple tasks need the CPU at the same time
Only *one* thread can run on one CPU core at any moment.
So the OS switches between threads to give everyone a chance.

Example:
Your laptop running:

- Chrome
- VS Code
- Spotify
- Background processes

The OS switches between them super fast → you feel like they run simultaneously.

—>A thread is waiting for something
For example:

- Waiting for a network response
- Waiting for user input
- Waiting for disk I/O

Instead of wasting time, the CPU switches to another thread.

—>Priority decisions

High-priority tasks (like system tasks) interrupt low-priority ones.

When Does Context Switching Happen?

1. Time Slice Expired (Round Robin Scheduling)
If a thread's allotted CPU time is over → switch.

2. I/O Operations
If a thread performs:

- database query
- API call
- file read
  → it pauses → CPU switches

3. Interrupts
Hardware interrupts (keyboard, mouse, network card) cause switching.

4. Thread/Process Creation
New threads/processes get scheduled → switching occurs.

Conclusion:

Is Context Switching good or bad?
Good → allows multitasking
But...
Too much context switching = performance drop
Because saving/restoring context has overhead.

REAL TIME Example : Instagram

When you upload a photo:

- UI thread shows progress
- Worker thread uploads file
- Another thread fetches new posts

Context switching keeps the app responsive.

Concept	Meaning
What	Saving state of one thread → switching to another
Why	To run multiple tasks and keep system responsive
When	Time slice over, I/O wait, new task, interrupts
How	OS saves registers, stack pointer, program counter