🧩 Scheduling & Context Switching

> “Fairness, efficiency, and the illusion that everyone’s program is special.”

💡 Big Idea

The CPU is the most coveted resource in the system.
Scheduling decides **who gets it**, **for how long**, and **when to swap them out**.
Context switching is how that swap actually happens.

⚙️ Goals of CPU Scheduling

- **Maximize** CPU utilization
- **Minimize** waiting time and turnaround time
- **Maintain fairness** among processes
- **Respond quickly** to interactive users

The trick? You can’t optimize all four at once.

🧠 Types of Schedulers

Type	When It Runs	Role
Long-Term (Job)	When jobs enter system	Decides which jobs to admit
Short-Term (CPU)	Every quantum or I/O event	Chooses which process gets CPU
Medium-Term	Occasionally	Suspends or resumes processes (swapping)

🧩 Scheduling Criteria

Metric	Meaning
CPU Utilization	% of time CPU is busy
Throughput	Jobs completed per unit time
Turnaround Time	Submit → Finish
Waiting Time	Time spent in ready queue
Response Time	Time to first output

Different workloads → different priorities.

⚖️ Scheduling Algorithms at a Glance

Algorithm	Pre-emptive?	Best For
FCFS (First Come First Served)	No	Batch jobs with similar lengths
SJF (Shortest Job First)	No	Predictable CPU bursts
SRTF (Shortest Remaining Time First)	Yes	Interactive systems
RR (Round Robin)	Yes	Time-shared systems
Priority	Either	Controlled scheduling
Multilevel Queue	Either	Mixed workloads

🧩 Round Robin Example

Quantum = 4 ms

```
P1: 0–8
P2: 0–4
P3: 0–2
```

Timeline:

```
|P1|P2|P3|P1|P1|
0      4  6     14
```

Fairness ↑, context switch overhead ↑.

🧠 Priority Scheduling

- Each process gets a priority value.
- Highest priority runs first.
- Can be pre-emptive or non-pre-emptive.

Problem: Starvation of low-priority tasks.

Fix: *Aging* — gradually increase priority over time.

🧩 Multilevel Queue

Separate ready queues for different types of processes:
interactive, batch, system, etc.

Each queue has its own scheduling algorithm.
High-priority queues pre-empt lower ones.

Great for OS design; terrible for fairness if mis-tuned.

🧠 Context Switching Explained

When the CPU switches from one process to another:

1. Save the old process's state (registers, PC, etc.)
2. Load the new process's state
3. Update CPU and MMU to reflect the new process

Takes microseconds → millions per second possible.

⚙️ Context Switch Overhead

Factor	Impact
Saving & restoring registers	CPU cycles lost
Cache flush / reload	Miss penalty
TLB invalidation	Memory latency
OS scheduler time	Administrative tax

Too many switches → thrashing.

🧩 Pre-emption vs Co-operative

- **Pre-emptive:** OS forcibly interrupts a running process.
  > Keeps system responsive.
- **Co-operative:** Process voluntarily yields CPU.
  > Simpler but dangerous if a process misbehaves.

Modern OSes = mix of both.

🧠 Scheduling in Practice

System	Strategy
Linux CFS	Fair proportion of CPU time per task (based on weight)
Windows NT	Priority-based round robin per class
Mac / BSD	Multilevel feedback queue with aging
RTOS	Strict priority + deadline awareness

🧩 Visualization Idea – Scheduler Loop

Simulate ready queue rotation for 3-5 jobs.

Track `ready → running → waiting → ready` states.

Display timeline or Gantt chart of CPU allocation.

Pairs well with the **Process Zoo** project.

🎯 Exit Prompt

> “What does ‘fair’ mean to a CPU scheduler?”
> Discuss whether fairness is about equal time, equal progress, or equal importance.

💾 Summary

Concept	Core Idea
Scheduling	Decides who runs and for how long
Context Switch	Mechanism that makes it happen
Algorithms	Balance throughput vs responsiveness
Overhead	Unavoidable, manageable
Fairness	Philosophy disguised as code

🧭 Next Topic

Week 01 Topic 04: Concurrency & Synchronization ▶️

Week 01 Topic 04: Concurrency & Synchronization ▶️

📚 References & Credits

> **topic:** "Operating Systems Week 1 Lecture Slides"
> **focus:** "Scheduling and Context Switching"
> **format:** "Markdown-based slide content"
> **author:** "T. Griffin and OpenAI GPT-5"
> **credit:** "Concept scaffolding with ChatGPT (OpenAI GPT-5)"