Operating Systems – Week 4 Lecture Notes

Instructor: SDB
Theme: Execution Management
Topic: *Context Switching and Scheduler View*

Lecture Script

Welcome to Week 4!

Today we explore a critical internal OS mechanism: **context switching** — the ability of the OS to suspend one process and resume another seamlessly.

Without context switching, your computer would run only one app at a time. Open Chrome, and Word freezes. Run Zoom, and your file sync halts. Clearly not acceptable.

We also dissect the **scheduler's role**, design goals, and how it views and prioritizes processes.

Core Concepts and Definitions

❖ *Context Switch*

Definition:
A **context switch** is the act of storing the state (context) of a running process so the CPU can load and resume the execution of another.

It allows **multitasking** by enabling the CPU to be shared across many processes.

❖ *When Does a Context Switch Happen?*

- Time slice expires (preemptive scheduling)
- Higher priority process arrives
- Process blocks on I/O
- System call triggers a yield or wait
- Interrupt handling

❖ *What Is Saved During a Context Switch?*

Component	Purpose
CPU registers	Resume computation
Program counter (PC)	Track next instruction
Stack pointer	Manage function calls
Scheduling metadata	Priority, state
Memory mappings	Virtual memory context (via MMU)

These are stored in the process's **PCB** and restored during reactivation.

❖ *Dispatcher*

Definition:
The **dispatcher** is the OS component responsible for:

- Loading the selected process context
- Resuming its execution by jumping to PC
- Ensuring correct memory mapping and stack

It works as the last stage of the scheduling decision.

❖ *Scheduler: A Conceptual View*

- Maintains **Ready Queue** and **Waiting Queue**
- Chooses next process to run based on policy (FCFS, Round-Robin, etc.)
- May use **priority, fairness, deadlines, or aging**
- Manages **CPU-bound** and **I/O-bound** process balance

Modern schedulers also account for:

- Affinity (bind to CPU core)
- Load balancing
- Energy awareness (on mobile)

Caselet: Zoom + File Sync + Spotify

Scenario:

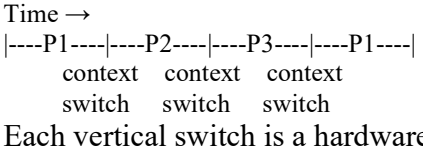
- Zoom (real-time audio/video)
- File Sync (OneDrive/Dropbox in background)
- Spotify (continuous audio)

CPU Switch Decisions:

- Zoom gets priority due to low-latency needs
- File Sync is I/O-bound → often in Waiting
- Spotify is preempted occasionally

Each context switch occurs in ~1–5µs.

Visualization of Context Switch Timing



Each vertical switch is a hardware/software interrupt triggering a switch in the kernel.

In-Class Shell Demo

```
time ./heavy_process        # Observe user vs system vs real time
chrt -p $$                  # Show process scheduling priority and policy
cat /proc/sched_debug       # Advanced: Linux scheduler metrics (CFS stats)
```

Optional:

- Use `top` → press `H` to show threads
- Observe frequent context switches on multi-threaded apps

Glossary of Terms

Term	Definition
Context Switch	Switching CPU from one process to another
Scheduler	OS module selecting which process runs next
Dispatcher	Loads process context into CPU
Preemption	Forcibly interrupting a process to run another
Affinity	Binding a process to specific CPU cores
CFS	Completely Fair Scheduler used in Linux

Exploration Topics

- Thread vs process context switches (cheaper?)
- How Linux handles context switch overhead (`perf`, `/proc`)
- Investigate `sched_setaffinity` and `nice`
- Compare time slices in real vs embedded OSes
- Explore multi-core scheduling strategies

☑Summary

- Context switching is essential for multitasking
- The PCB stores all necessary process state
- The **dispatcher** performs the low-level switch
- The **scheduler** decides who gets the CPU and when
- Frequent context switches increase responsiveness but may reduce cache locality

---

✍ Review & Exercises

1. List the steps involved in a context switch
2. What fields in the PCB are used during a switch?
3. Write a small C program with 2 forked processes and observe their CPU usage
4. What's the overhead of a context switch? How can we measure it?
5. Observe context switches in `top` or `vmstat`
6. Challenge: Implement a cooperative scheduler using `setjmp/longjmp` in user space