OPERATING SYSTEMS & CONCURRENT PROGRAMMING IN ACTION

Part 1 of 11: Seminar Introduction Speech

Welcome to our seminar entitled *"Operating Systems & Concurrent Programming in Action."* In this first session, we will set the stage for the entire series and outline the primary themes we will address, including concurrent programming, resource management, deadlocks and avoidance, as well as system monitoring and safety algorithms. Throughout this journey, we will demonstrate how these core operating system (OS) concepts can be practically implemented using Bash scripts. All relevant scripts discussed are publicly available in the following GitHub repository: https://github.com/antonioclim/OS5.

Despite the academic nature of this seminar, our goal is to explain these principles in a manner suitable for newcomers—fostering a foundational understanding without assuming extensive prior knowledge. Let us begin by briefly introducing the key topics:

---

1. Concurrent Programming

Operating systems often support the concurrent execution of multiple processes (or threads), which can vastly improve the efficiency and responsiveness of applications. Concurrency, however, requires careful coordination, because unsynchronised processes may inadvertently interfere with one another, causing data inconsistencies or even system crashes.

A classic illustration of this challenge is the "Dining Philosophers Problem," where each philosopher around a table needs two forks to eat but shares them with neighbours. Without proper synchronisation, all philosophers may hold one fork and wait forever, causing a system-wide standstill (deadlock). In upcoming sessions, we will explore how practical Bash scripts leverage synchronisation mechanisms (e.g., file locking via flock) to solve these kinds of concurrency issues in an operating system environment.

---

2. Resource Management

An operating system's role in resource management is to efficiently allocate limited resources—such as CPU cycles, memory, and storage—to various processes. Through this seminar, you will see how Bash scripts can dynamically monitor CPU load, memory usage, and disk utilisation, giving system administrators immediate insight into system performance.

Practical resource management extends to handling temporary files and processes that remain in memory even after completion (so-called "zombie processes"). You will learn how Bash scripts can safely clean up orphaned system resources, prevent accumulation of unnecessary data, and ultimately maintain system stability.

---

3. Deadlocks and Avoidance

Deadlocks occur when processes hold resources while simultaneously waiting for resources that other processes have locked, leading to an inescapable cycle. These incidents can cripple a system if not detected or prevented. In this seminar, we will:

- **Observe** intentional deadlock scenarios through Bash scripts that highlight how simple resource requests can inadvertently lead to circular waiting.
- **Detect** potential deadlock conditions with command-line tools such as **lsof**, which identifies which processes hold certain resources.
- **Prevent** system-wide lockups via advanced locking strategies, including timeouts and retries.

By witnessing these mechanisms in real scripts, you will gain actionable insights into how to handle deadlocks proactively and maintain a stable operating environment.

---

4. System Monitoring and Safety Algorithms

Finally, we will introduce *system monitoring* techniques and delve into the Banker's Algorithm—a foundational safety-checking method formulated by Edsger Dijkstra. The Banker's Algorithm evaluates resource requests in advance to ensure the system never enters an *unsafe* state (where there is no sequence of allocations that would allow every process to finish without conflict).

We will demonstrate Bash scripts that simulate multiple processes making resource requests and show how the Banker's Algorithm can either grant or deny these requests to sustain overall system safety. Effective logging strategies for tracking resource usage will also be presented, helping to maintain clarity in potential troubleshooting or forensic analysis.

Seminar Roadmap

Across the remaining parts of this seminar, each concept—concurrency, resource management, deadlocks, and safety algorithms—will be explored in-depth, reinforcing theory with practical Bash implementations. By the conclusion of our 11-part series, you should be able to:

1. **Grasp the Principles of Concurrent Programming**
   Understand how to design processes that run in parallel without risking data corruption or inconsistent states.
2. **Manage System Resources Effectively**
   Use Bash scripting to monitor, allocate, and clean up CPU time, memory, file systems, and other critical resources.
3. **Detect and Prevent Deadlocks**
   Anticipate potential cycles of resource contention and apply avoidance or resolution strategies to keep systems running smoothly.
4. **Apply Safety Algorithms (e.g., Banker's Algorithm)**
   Ensure that processes acquire resources only when the system's overall state remains safe, preventing unrecoverable lockups.

Throughout, we will consistently emphasise practical, example-driven learning. Each script showcased will be accessible for download and further exploration, allowing you to reproduce and adapt these techniques in your own environments.

Thank you for joining us. Let us now embark on our exploration of *Operating Systems & Concurrent Programming in Action*, starting with the foundations presented today. Your engagement here will build toward deeper insights in subsequent sessions, where these concepts become ever more tangible through hands-on Bash scripting.

References

- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts* (10th ed.). Wiley. https://doi.org/10.1002/9781119320913

End of Part 1 of 11