Operating System Topics

- **What is this?**

It’s a comprehensive list of OS topics (~95% total coverage or a +1000 page book) made for experimentation and learning through it.

- **How Can I Use It?**

Search for the answer, explanation and meaning from any legit source you know, that’s it.

Active searching is better than passive reading.

“Jetbrains corp, chill out”

Made by @maphy01 on Telegram

STAGE 1

ABSOLUTE FUNDAMENTALS

Objectives

- ○ Process vs Program
- ○ Kernel Space vs User Space
- ○ System Call Mechanism
- ○ Hardware Abstraction Layer
- ○ BIOS/UEFI Boot Process
- ○ Shell Fundamentals
- ○ Microkernel vs Monolithic Kernel
- ○ ACPI Power States (S0-S5)
- ○ Memory Hierarchy (Memory Pyramid) (Registers → Cache → RAM → Disk)

Side Missions

- ○ Why must all I/O eventually involve the kernel?
- ○ How does CPU ring architecture enforce isolation?
- ○ What happens physically during a context switch?

STAGE 2

CORE MECHANICS

Objectives

- ○ Interrupt Handling (IRQs)
- ○ Process States (Ready/Running/Blocked)
- ○ Thread Implementation Models
- ○ Memory Protection Units
- ○ Bootloaders (GRUB/systemd-boot)
- ○ Device Driver Types (Char/Block/Net)
- ○ POSIX Standards
- ○ Synchronous vs Asynchronous I/O
- ○ Secure Boot
- ○ Dynamic Voltage/Frequency Scaling (DVFS)

Side Missions

- ○ Why can't user processes disable interrupts?
- ○ How do kernel bypass techniques violate OS principles?
- ○ Explain how a keyboard press becomes a process event.

STAGE 3

RESOURCE VIRTUALIZATION

Objectives

- Virtual Memory Architecture
- Paging vs Segmentation
- Page Replacement Algorithms
- File System Inodes & Dentries
- Disk Scheduling (Elevator/C-SCAN)
- Loadable Kernel Modules
- IPC Methods (Pipes/Shared Mem/Message Queues)
- CPU Caches & Locality

Side Missions

- Why is TLB miss handling considered critical path?
- How do SSDs break traditional disk scheduling?
- Calculate worst-case thrashing scenario for LRU.

STAGE 4

CONCURRENCY ARCHITECTURE

Objectives

- Preemption vs Cooperative Multitasking
- Lock Implementations (Spinlocks/Mutexes)
- Deadlock Conditions & Mitigations
- Reader-Writer Problem Solutions
- Kernel Preemption Models
- RCU Synchronization
- Timer Subsystems (jiffies/HRTimers)
- Atomic Operations
- Real-Time Schedulers (EDF/Rate-Monotonic)

Side Missions

- Why can't priority inheritance solve all priority inversions?
- How do memory barriers prevent speculative execution chaos?
- Design lock-free producer/consumer for 64-core NUMA.

STAGE 5

STORAGE & FILESYSTEMS

Objectives

- Journaling vs Copy-on-Write FS
- RAID Configurations
- Virtual Filesystem Abstraction
- Storage Stack (Block Layer/IO Scheduler)
- Page Cache Writeback
- Distributed FS Consistency Models
- Storage Area Networks
- Persistent Memory Programming

Side Missions

- Why does ext4 use delayed allocation despite data loss risk?
- How does Ceph achieve POSIX compliance without POSIX?
- Quantify fsync() performance tradeoffs for databases.

STAGE 6

NETWORKING & DISTRIBUTION

Objectives

- Network Stack Layers (L2-L7)
- Socket Implementation
- Protocol Offloading (TSO/LRO)
- Network Namespaces
- Firewalling (Netfilter/BPF)
- RDMA & Zero-Copy Networking
- Distributed Shared Memory
- Consensus Protocols in OS Kernels
- eBPF for Tracing/Observability

Side Missions

- Why does the Linux kernel avoid TCP segmentation?
- How do eBPF programs safely modify live packet flows?
- Explain time synchronization attacks in distributed systems.

STAGE 7

SECURITY & ISOLATION

Objectives

- Capability-Based Security
- Memory Protection Keys
- TPM Integration
- Kernel Hardening (KASLR/SMEP)
- Mandatory Access Control
- Hypervisor-Assisted Security
- Side-Channel Attack Mitigations
- Kernel Exploit Patterns (ROP/JOP/Heap Spray)
- Trusted Execution Environments (SGX/TrustZone)

Side Missions

- Why can't ASLR protect against all ROP attacks?
- How do transient execution attacks bypass hardware isolation?
- Design secure IPC for mutually distrustful processes.

STAGE 8

DEBUGGING, PROFILING & LEGACY

Objectives

- Kernel Debuggers (kgdb/kdb)
- Tracing Systems (ftrace/perf/eBPF)
- Crash Dump Analysis
- Binary Compatibility Layers (WINE/ROSSETA)
- Legacy Driver Virtualization
- Live Patching (kpatch/kgraft)

Side Missions

- Why can't ASLR protect against all ROP attacks?
- How do transient execution attacks bypass hardware isolation?
- Design secure IPC for mutually distrustful processes.

STAGE 9

VIRTUALIZATION & CLOUD

Objectives

- Hypervisor Types (Type 1/2)
- Paravirtualization
- Container Runtimes
- Unikernel Tradeoffs
- Virtual Machine Introspection
- Hardware Virtualization (VT-x/AMD-V)
- Live Migration
- Virtual Machine Introspection (VMI)
- Nested Virtualization

Side Missions

- Why can't VMs achieve 100% native performance?
- How do container escapes exploit kernel subsystems?
- Design NUMA-aware VM scheduling for mixed workloads.

STAGE 10

AI/ML SYSTEMS FOUNDATIONS

Objectives

- GPU Memory Management (UVA)
- Asynchronous I/O Optimization (io_uring)
- Kernel Bypass for ML (RDMA)
- NUMA Scheduling for ML
- Huge Pages & TLB Pressure
- Persistent Memory Checkpointing
- Quantized OS Services
- ML Model Security Enclaves
- Heterogeneous Auto-Tuning (CPU/GPU/TPU)

Side Missions

- Why does PyTorch prefer fork() over pthreads?
- How does NCCL avoid kernel bottlenecks in all-reduce?
- Design page fault handler for sparse model weights. (Concept on the Paper “CoP”)

STAGE 11

FUTURE AI/ML-OS INTEGRATION

Objectives

- Heterogeneous Scheduling (CPU/GPU/TPU)
- In-Network Computation Offloading
- Model-Aware Paging
- Federated Learning OS Services
- Optical Compute Interconnects
- Memristor-Based Storage Class Memory
- Kernel-Embedded Inference Runtimes
- Quantum Coprocessor Abstraction

Side Missions

- Could attention mechanisms replace page replacement algorithms? Why?
- Design OS support for dynamic precision switching in ML accelerators. (CoP)
- How would OS manage non-von Neumann compute fabrics?