Guide2Code - Operating System (OS) Roadmap 🖥️

🟢 Phase 1: Beginner Level

📚 Topics to Learn:

1. Introduction to Operating Systems (What is an OS? Types of OS)
2. Process Management (Processes, Threads, Scheduling)
3. Memory Management (Paging, Segmentation, Virtual Memory)
4. File System Basics (File Structure, Directories, Permissions)
5. Basics of I/O Management (Device Drivers, Interrupts)
6. Deadlocks (Causes, Prevention, Detection, Recovery)
7. Inter-Process Communication (Pipes, Shared Memory, Message Passing)
8. Basics of Synchronization (Mutex, Semaphores, Critical Section)
9. Basics of Shell Scripting (Linux Commands, Bash Scripting)
10. CPU Scheduling Algorithms (FCFS, SJF, RR, Priority Scheduling)

🛠️ Beginner Project Ideas:

- **Simple Shell Script** – Automate basic tasks using Bash scripting
- **Process Scheduler Simulator** – Implement FCFS, SJF, and RR scheduling
- **Memory Management Visualizer** – Demonstrate paging and segmentation
- **File Permissions Checker** – Analyze and modify file permissions
- **Basic Task Manager** – List and manage system processes

🟡 Phase 2: Intermediate Level

📚 Topics to Learn:

1. Advanced Process Synchronization (Monitors, Readers-Writers Problem)
2. Advanced Memory Management (TLB, Frame Allocation, Demand Paging)
3. File System Implementation (Inodes, Journaling, RAID)
4. Virtualization Basics (Hypervisors, Containers, VMs)
5. System Calls & Kernel Mode (User vs Kernel Mode, API Calls)
6. Disk Scheduling Algorithms (FCFS, SSTF, SCAN, C-SCAN)
7. Linux Kernel Basics (Modules, System Calls, File System)

8. Advanced Shell Scripting (Loops, Functions, Automation)
9. Introduction to Security in OS (Access Control, Malware, Encryption)
10. Case Studies (Windows vs Linux vs macOS Architecture)

🛠 Intermediate Project Ideas:

- **Thread Synchronization Tool** – Implement producer-consumer problem
- **Custom File System Simulator** – Design a simple file system structure
- **Virtual Memory Manager** – Implement page replacement algorithms
- **Disk Scheduling Visualizer** – Simulate different disk scheduling algorithms
- **Basic OS Kernel Module** – Write a simple Linux kernel module

🔴 Phase 3: Advanced Level

📚 Topics to Learn:

1. Advanced Kernel Development (Process Scheduling, Device Drivers)
2. Distributed Operating Systems (Clusters, Distributed File Systems)
3. Advanced Virtualization Concepts (Paravirtualization, KVM, Xen)
4. Real-Time Operating Systems (RTOS, Applications, Case Studies)
5. OS Security & Cryptography (Firewalls, Secure Boot, Sandboxing)
6. OS for Embedded Systems (Android Kernel, IoT OS)
7. File System Performance Optimization (Caching, Defragmentation)
8. Cloud-Based OS Concepts (Serverless Computing, OS-Level Virtualization)
9. Fault Tolerance & Recovery Mechanisms (Checkpointing, Redundancy)
10. AI & Machine Learning in OS (Predictive Scheduling, Smart Resource Allocation)

🛠 Advanced Project Ideas:

- **Custom Mini Operating System** – Build a basic OS with bootloader
- **Distributed File System** – Implement file sharing across multiple nodes
- **OS-Level Virtual Machine Monitor** – Create a basic hypervisor
- **Secure Sandboxed Environment** – Develop an isolated execution space
- **Process Monitoring & Optimization Tool** – Enhance system resource management