OPERATING SYSTEM

User Mode and Kernel Mode

A **computer operating system** works in **two distinct modes** to protect itself and manage system resources efficiently:

1. **User Mode**
2. **Kernel Mode**

These modes define **what the CPU can and cannot do**.

1. User Mode

Definition

User Mode is a restricted mode in which applications run.
Programs in user mode cannot directly access hardware or critical system resources.

Characteristics

- Limited privileges for safety
- Cannot execute privileged instructions
- Must request OS services via **system calls** to access hardware
- Example programs: web browsers, word processors, games

2. Kernel Mode

Definition

Kernel Mode (also called **Supervisor Mode or System Mode**) is a **privileged mode** in which the **operating system kernel runs**.

Characteristics

- Full access to CPU instructions and hardware
- Can execute any operation, including I/O, memory, and device management
- Runs system calls and device drivers
- Example: handling `fork()`, reading files, accessing network card

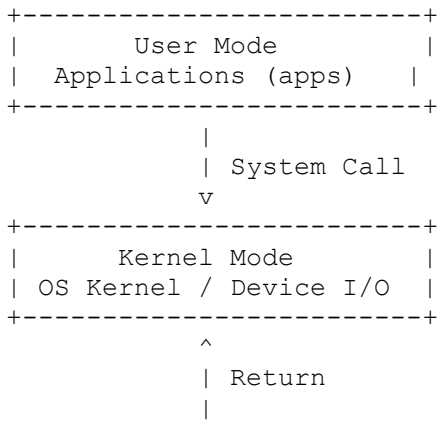
Key Differences

Feature	User Mode	Kernel Mode
Access Level	Restricted	Full
Execution	Applications	Operating System
Hardware Access	Indirect (via system calls)	Direct
Privileges	Cannot execute sensitive instructions	Can execute all instructions
Examples	Browser, Word Processor	`fork()`, file system operations, device drivers

Mode Switching

- When a program in **user mode** needs to access hardware or OS resources, it **makes a system call**.
- The CPU **switches to kernel mode**, executes the request, then returns to user mode.
- This ensures **security and stability** of the system.

Diagram



1. Why Mode Switching is Necessary

Mode switching allows a CPU to move safely between **User Mode** and **Kernel Mode**.

Reasons:

1. **Security**
    - User programs cannot directly access hardware or system-critical resources.
    - Prevents accidental or malicious damage to the system.
2. **Stability**
    - Errors in user programs don't crash the OS.
    - Only kernel code can perform sensitive operations safely.
3. **Controlled Access to Hardware**
    - Hardware access must go through OS to prevent conflicts.
    - Example: Reading/writing a file or sending data to a printer.
4. **Resource Management**
    - OS ensures fair sharing of CPU, memory, and devices.
    - Mode switching allows OS to enforce rules.

2. Examples of Instructions Allowed in Kernel Mode but Not in User Mode

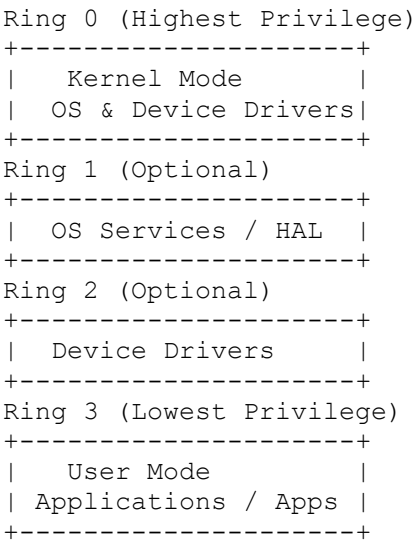
Instruction / Operation	Allowed in Kernel Mode?	Allowed in User Mode?	Reason
Direct I/O access (read/write hardware ports)	✅ Yes	❌ No	Could corrupt devices if done directly
Disable or enable interrupts	✅ Yes	❌ No	Could make system unresponsive
Set memory management registers (like page tables)	✅ Yes	❌ No	Could bypass memory protection

Instruction / Operation	Allowed in Kernel Mode?	Allowed in User Mode?	Reason
HALT CPU or reboot system	✅ Yes	❌ No	User program shouldn't stop system
Modify kernel data structures	✅ Yes	❌ No	Critical for OS integrity

Rule: Only **privileged instructions** can run in kernel mode.

3. CPU Privilege Rings Diagram

Many modern CPUs (like x86) use **rings** to enforce privilege levels.



- **Ring 0** → Full access (kernel)
- **Ring 3** → Limited access (user programs)
- System calls are the **gateway from Ring 3 → Ring 0**.