Lecture#03 System Calls in O.S

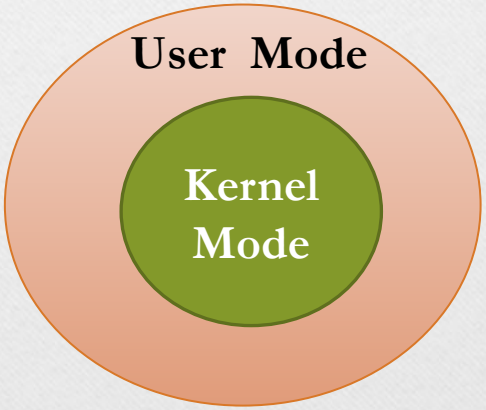
An Overview of System Calls and Their Importance

Definition

- System calls provide an interface between a program (user space) and the operating system (kernel space). They enable a program to request services from the operating system's kernel, such as managing hardware devices, memory, files, and processes.

System Calls

- User Mode:
  - Safer Mode
  - Need access to resources
  - Switch from User-to-Kernel Mode.
- Kernel Mode:
  - Privileged Mode
  - Direct access to resources
  - If process crash during execution, whole system will crash



Why System Calls are Needed

- 1. Resource Access: Controlled access to hardware resources.
- 2. Abstraction: Simplifies hardware operations for developers.
- 3. Security: Prevents unauthorized or harmful actions by programs.

Categories of System Calls

- 1. Process Control: Manage processes, Load process from secondary to main memory (e.g., abort(), wait(),fork(), exec()).
- 2. File Management: Handle files (e.g., open(), read(), write(), close(), Create()).
- 3. Device Management: Interact with hardware devices (e.g., read(), reposition(), ioctl()).
- 4. Information Maintenance: Retrieve system/process info, attributes, get system time, date etc (e.g., getpid(), getppid()).
- 5. Communication: Facilitate inter-process communication (e.g., pipe()).

System Call Working Mechanism

- 1. Request: User program invokes a system call.
- 2. Mode Switch: CPU switches to kernel mode.
- 3. Execution: Kernel performs the requested task.
- 4. Mode Switch Back: CPU returns to user mode and sends result to program.

Types of System Calls

- 1. Blocking System Calls: Program waits for operation completion (e.g., read()).
- 2. Non-Blocking System Calls: Program continues execution without waiting (e.g., poll()).

Examples of System Calls

- - Process: fork(), exit()
- - File Management: open(), read(), write()
- - Device Management: ioctl(), read(), write()
- - Communication: pipe(), send(), recv()
- - Information Maintenance: getpid(), alarm()

Advantages of System Calls

- 1. Efficiency: Enables direct communication with the kernel.
- 2. Security: Ensures controlled access to resources.
- 3. Abstraction: Simplifies interaction between applications and hardware.

System Call Interfaces

- 1. High-level APIs: Wrappers like POSIX simplify system call usage.
- 2. Direct System Calls: Programs invoke kernel services directly.

Examples in Operating Systems

- - Windows: API calls like CreateProcess, ReadFile.
- - Linux/Unix: System calls like fork(), exec(), open().