Lecture Notes: Operating Systems - Simplified : Lec 5

# Introduction

In operating systems, applications interact with the kernel to perform operations such as file management, process management, and device communication. These interactions are performed using system calls, which are mechanisms for requesting kernel services.  
System calls allow user programs to perform actions that they wouldn't normally have permission to do, like accessing hardware or communicating with other processes. The transition from user space to kernel space is typically done through software interrupts.

# How Applications Interact with Kernel

1. Applications interact with the kernel using system calls. For example, the 'mkdir' command to create a directory is just a wrapper for the actual system call that interacts with the kernel.  
  
Example: mkdir 'laks'  
- The 'mkdir' command indirectly calls the kernel and asks the file management module to create a new directory.  
- 'mkdir' is a wrapper around system calls.  
- It interacts with the kernel using system calls.  
  
2. When creating a process, the user executes a process in the user space (US). It then gets a system call and switches to kernel space (KS) to execute the process creation system call.  
  
Example:  
- The user executes a process in user space.  
- It triggers a system call.  
- The kernel executes the system call to create a process.  
- The control returns to user space.  
  
Transitions from user space to kernel space are done through software interrupts.

# Types of System Calls

System calls are implemented in C and categorized into different types based on their functionality. Here are some types of system calls:

## 1) Process Control

- End, abort  
- Load, execute  
- Create process, terminate process  
- Get process attributes, set process attributes  
- Wait for time  
- Wait event, signal event  
- Allocate and free memory

## 2) File Management

- Create file, delete file  
- Open, close  
- Read, write, reposition  
- Get file attributes, set file attributes

## 3) Device Management

- Request device, release device  
- Read, write, reposition  
- Get device attributes, set device attributes  
- Logically attach or detach devices

## 4) Information Maintenance

- Get time or date, set time or date  
- Get system data, set system data  
- Get process, file, or device attributes  
- Set process, file, or device attributes

## 5) Communication Management

- Create, delete communication connection  
- Send, receive messages  
- Transfer status information  
- Attach or detach remote devices

# Differences between Software Interrupts and System Calls

Although both system calls and software interrupts involve interactions with the kernel, they differ in purpose and how they are triggered.

## System Calls:

- A system call is a deliberate request by a program to perform a specific operation that it doesn't have permission for, such as accessing hardware or files.  
  
Example: A program calling the 'open()' system call to read a file.  
  
System calls provide controlled access to kernel services and are planned, with the program explicitly requesting the service.

## Software Interrupts:

- A software interrupt is an unexpected event triggered by hardware or a program that requires immediate attention from the kernel.  
  
Example: A division by zero triggers a software interrupt to handle the error.  
  
Software interrupts are unplanned and happen when something unexpected occurs, like an error or hardware event.

Key Differences:  
1. System calls are intentional requests for services, while software interrupts are reactions to events that need immediate handling.  
2. System calls are executed when a program asks for specific tasks, but software interrupts are triggered by hardware events or errors.  
3. System calls are used for planned operations like reading a file, while software interrupts handle emergencies, like errors.