OPERATING SYSTEM

System Calls – Definition & Types

Definition of System Calls

A **system call** is a special request made by a program to the operating system **to perform tasks that the program itself cannot do directly**, such as accessing hardware, creating a process, reading files, or getting system information.

System calls act as an **interface between user-level programs and the operating system kernel**.

Types of System Calls (Defined & Explained)

System calls are grouped into **five major categories**:

1. Process Control System Calls

Definition

These system calls are used to create, manage, and terminate processes in the operating system.

Examples

- `fork()` – Create a new process
- `exec()` – Run a new program
- `exit()` – Terminate a process
- `wait()` – Wait for a process to finish
- `getpid()` – Get process ID

2. File Management System Calls

Definition

These calls allow programs to **create, read, write, and delete files** and directories.

Examples

- `open()` – Open a file
- `read()` – Read data
- `write()` – Write data

- `close()` – Close a file
- `create()` / `delete()` – Manage files

3. Device Management System Calls

Definition

These system calls are used to **interact with hardware devices** such as printers, disks, keyboards, USB devices, etc.

Purpose

- Allocate and free devices
- Transfer data to/from devices
- Control device behavior

Examples

- `ioctl()` – Control or configure a device
- `read()` – Read from a device
- `write()` – Write to a device
- `open()` / `close()` – Use or release a device

4. Information / Information Maintenance System Calls

Definition

These system calls are used to **retrieve or modify system-related information**, such as time, process ID, user ID, and system status.

Examples

- `getpid()` – Get process ID
- `getppid()` – Get parent process ID
- `uname()` – Get system information
- `time()` – Get current time
- `getuid()` – Get user ID

5. Communication System Calls (IPC – Inter-Process Communication)

Definition

These calls allow processes to **communicate with each other** through shared memory, message passing, or network communication.

Examples

- `pipe()` – Create a communication pipe
- `shmget()` – Allocate shared memory
- `send()` / `recv()` – Sending/receiving messages
- `socket()` – Create a network socket