System Call:

Operating system has lot of responsibilities and functions to perform and is considered to be the busiest part of a computer system. Sometimes it even gets worse as multiple requests and tasks are required to be manipulated at the same time. **SYSTEM CALL** acts as a boon in doing all of these with ease.

System calls in an operating system act as the programmatic interface between a running application (user mode) and the operating system kernel (kernel mode). They enable user programs to request privileged services such as file I/O, process creation, and hardware access, ensuring system security and stability by managing hardware interactions.

Key Aspects of System Calls:

- Mode Switching: When a program invokes a system call, a "trap" instruction is executed, forcing the CPU to switch from User Mode to Kernel Mode to securely execute the request.
- Mechanism: Programs typically use APIs (like POSIX for Unix or Win32 for Windows) rather than direct system calls.

Types of System Calls:

- **Process Control**: Creates, terminates, and manages processes (e.g., fork(), exec(), exit(), wait()).
- **File Management**: Creates, reads, writes, and closes files (e.g., open(), read(), write(), close()).
- **Device Management**: Requests/releases devices (e.g., ioctl (), read(), write()).
- **Information Maintenance:** Transfers information between user program and OS, such as time or system data.
- Communications: Handles inter-process communication (IPC).

Common Examples (Unix/Linux): fork (), exec (), open (), read (), write (), kill (), wait ().

Without system calls, applications would have to manage hardware directly, leading to security breaches and inconsistent system behavior.

Note: It is sometime said as when the operating system is not capable to perform any specific activity using commands, it makes a SYSTEM CALL which in turn can make a tedious work look easier as per the execution is concerned.