Limited Direct Execution

Sanghyeok Park
School of Electrical and Electronics Engineering
electronics@cau.ac.kr
20234690

October 1, 2025

1 Introduction

This content is based on [1]. The operating system uses the technique of Limited Direct Execution to efficiently run programs while retaining control of the CPU. Limited Direct Execution means executing a program directly on the CPU as much as possible, but ensuring safety and control by relying on traps and system calls. Through the system call and trap mechanism, the OS enables safe transitions between user mode and kernel mode. When a program issues a system call or terminates, a trap occurs, the OS handles the request, and then returns control to the user program. By using a timer interrupt, the OS can regain control of the CPU even from non-cooperative processes. Finally, a context switch is a low-level mechanism that saves and restores registers and stack pointers, enabling the OS to switch execution between processes.

2 System Call and Trap Mechanism

User programs run directly on the CPU, but cannot perform privileged operations such as accessing the file system or managing memory. To safely allow these restricted operations, the operating system provides system calls and uses the trap mechanism. A trap transfers control from user mode to kernel mode, the OS performs the requested work, and then control is returned to the user program.

2.1 Trap

A trap occurs when a user-mode program requests a privileged operation or performs an illegal instruction. For example, operations such as file access, process creation, or memory allocation cannot be executed directly and must instead enter the OS via a system call. When a trap occurs, the hardware saves the current process' s register state onto its kernel stack, switches to kernel mode, and transfers control to the appropriate handler specified in the trap table.

2.2 Return-from-trap

Once the OS completes the requested system call or exception handling, it executes the `return-from-trap` instruction to return to user mode. During this step, the OS restores the saved registers and the PC from the kernel stack so that the program can resume execution seamlessly. Thus, the combination of trap and return-from-trap is the fundamental mechanism that allows the OS to temporarily gain control and then safely resume user-level execution.

2.3 Trap Table

The trap table is a data structure that stores the handler address for each trap number. During boot, the OS registers the location of the trap table with the hardware. Importantly, this registration requires a privileged instruction, and therefore cannot be executed in user mode.

3 Non-Cooperative Approach

3.1 Timer Interrupt

In a cooperative model, the OS only regains control of the CPU when a process makes a system call. However, if a malicious or buggy process enters an infinite loop and never calls the OS, the OS loses control. To solve this problem, modern systems use a timer interrupt. During boot, the OS programs the timer hardware to generate interrupts at fixed intervals. When a timer interrupt occurs, the currently running process is suspended, its state is saved, and the OS regains control to run its scheduler. This mechanism ensures that the OS always has the ability to run again on the CPU, enabling preemptive scheduling.

4 Context Switch

4.1 Saving and Restoring Context

A context switch occurs when the OS decides to stop running one process and resume another. This involves saving the current process's state and restoring the state of the process to be run.

- Save: The general-purpose registers, PC, and kernel stack pointer of the current process are saved into its process structure.
- Restore: The registers, PC, and kernel stack pointer of the next process are restored from its process structure.

4.2 Execution Flow

1. A timer interrupt occurs → hardware saves the registers of Process A onto its kernel stack.
2. The OS executes the trap handler → the scheduler selects Process B.
3. The `switch()` routine runs:
    - Save Process A's context into its process structure.
    - Restore Process B's context from its process structure.
    - Switch the stack pointer to Process B's kernel stack.
4. The OS executes `return-from-trap` → execution resumes in user mode at Process B's PC.

References

[1] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, 1.10 edition, November 2023.