Operating System
Lecture 5

Dr. Ghada Fathy

Table of contents

Operating System Operations

- **Interrupt driven** (hardware and software)

  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap):**

    - ❑ Software error (e.g., division by zero).
    - ❑ Request for operating system service.
    - ❑ Other process problems include infinite loops, processes modifying each other, or the operating system.

Operating System Operations

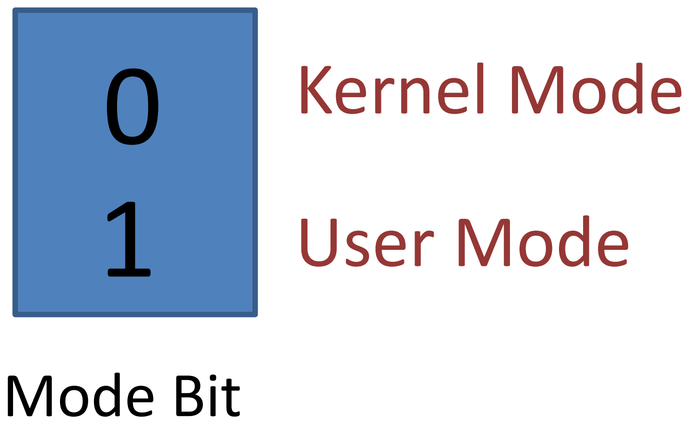
- **Operating System Operations**, how the **Operating System (OS)** protects itself and the computer using **different operating modes**.

  - Dual-Mode Operation

  - The CPU can work in two modes:

    1- User mode:
    - Normal programs (like apps) run here.
    - hey have limited access to system resources for safety.

    2- Kernel mode:
    - The operating system runs here.
    - It has full access to all hardware and memory.



  - Mode Bit
    - This is a special signal (bit) in the hardware that tells whether the CPU is in user mode or kernel mode.
    - It helps the system distinguish between normal user code and powerful system code.
    - Some commands (called **privileged instructions**) can only run in kernel mode — to prevent users from damaging the system.
    - When a program makes a system call (like opening a file), the CPU switches to kernel mod

Transition from User Mode to Kernel Mode

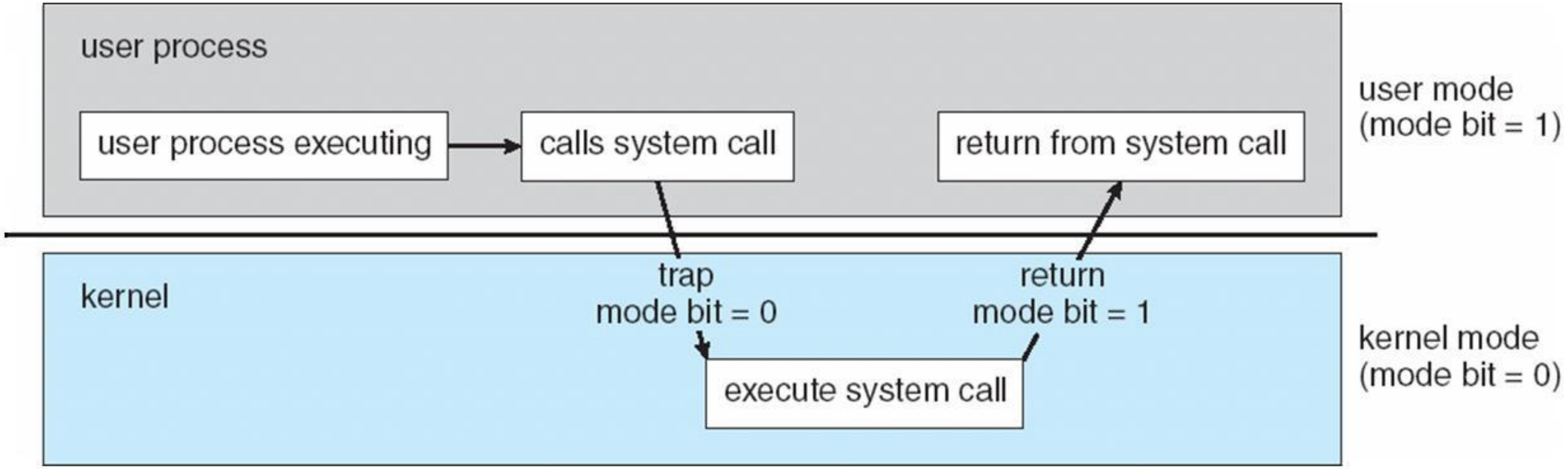
- The CPU **switches from user mode to kernel mode**, and how the system uses a **timer** to prevent programs from taking over the computer forever.

How it is work:

- The Operating System sets a counter (using a special, privileged instruction).
- This counter is automatically decreased by the clock as time passes
- When the counter reaches zero, an interrupt happens this forces the CPU to stop the current program and give control back to the OS.

The OS can then:

- Schedule another process, or
- Terminate the program if it's using too much time.

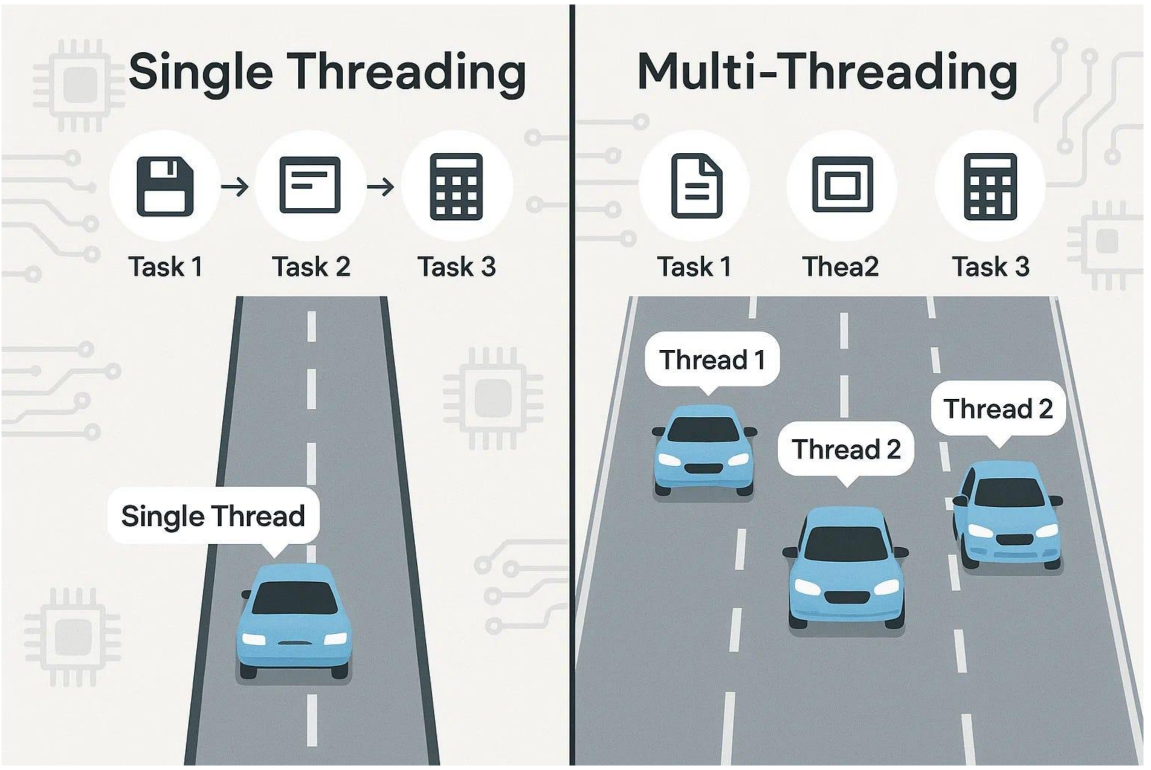

user process
user process executing
calls system call
return from system call
user mode
(mode bit = 1)
kernel
trap
mode bit = 0
return
mode bit = 1
execute system call
kernel mode
(mode bit = 0)

Process Management

A process is a program **in execution**. It is a unit of work within the system. Program is a ***passive entity***, process is an ***active entity***.

- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources

- Single-threaded process:
  - Has one program counter, meaning it runs one instruction at a time in sequence — like a single worker doing tasks one by one.
- Multi-threaded process:
  - Has multiple program counters — many threads doing different parts of the same job simultaneously, like a team of workers collaborating on one project.



Process Management

Many Processes at Once (Concurrency)

- Your system usually has **many processes** running together:
  - Some belong to **users** (like Chrome, Word, or a game 🎮).
  - Some belong to the **operating system** (background tasks that keep the system running).
- Even if you have **one CPU**, it feels like everything runs at once — that's because the OS uses **switching**:
  - It gives each process a tiny slice of CPU time, then quickly switches between them — super fast, so it looks simultaneous!

Process Management Activities

The **Operating System (OS)** acts like a **manager** 💼 — it controls how all programs (processes) run and work together smoothly.

- Creating and Deleting Processes
  - OS creates new processes when you open apps.
  - It deletes them when they finish or are closed.

    Example: Opening or closing Google Chrome.

- Suspending & Resuming Processes
  - The OS can pause a process to give time to another.
  - Later, it can resume it from where it stopped.

    Example: Pausing a video or download, then continuing later.

- Process Synchronization
  - Makes sure multiple processes don't interfere with each other.

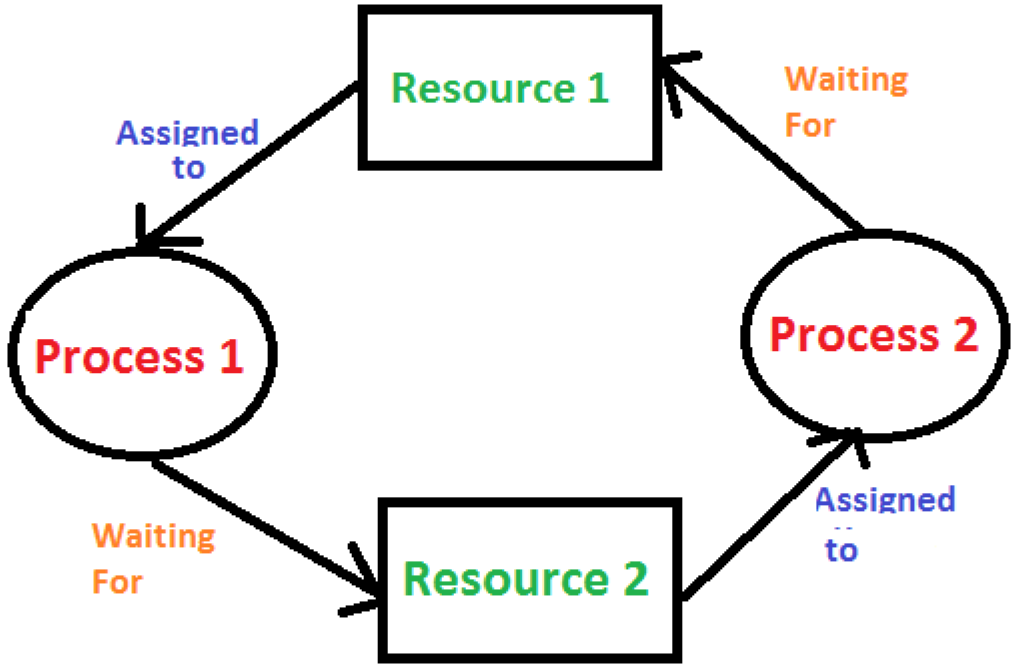
    Example: Two files trying to print at the same time — OS makes them take turns.

Process Management Activities

- Process Communication
  - Allows processes to share data and talk to each other safely.
    Example: A browser communicating with a network service to load a web page.
- Deadlock Handling
  - Prevents situations where processes get stuck waiting for each other forever.
    Example: Two cars blocking each other in a one-way street — OS helps one move first.

OS Handling Deadlock

The **Operating System (OS)** runs many processes at the same time. Sometimes, two or more processes get stuck waiting for each other this is called a **deadlock**.
The OS can handle deadlocks in four main ways:



1. **Deadlock Prevention**
   - The OS **designs the system** so that deadlocks **never happen**.
   - It avoids situations where a process holds one resource and waits for another.

2. **Deadlock Avoidance**
   - The OS **checks ahead of time** to see if a deadlock *might* happen.
   - If it detects a risk, it **won't allow** the action that could cause deadlock.

OS Handling Deadlock

3. **Deadlock Detection**
   - The OS **lets deadlocks happen**, but it **detects** them using an algorithm.
   - Then it **kills or restarts** one or more processes to fix the problem.

4. **Deadlock Ignorance**
   - The OS **ignores** the problem completely.
   - If a deadlock occurs, the system might just **restart**.


Thank you