Operating Systems

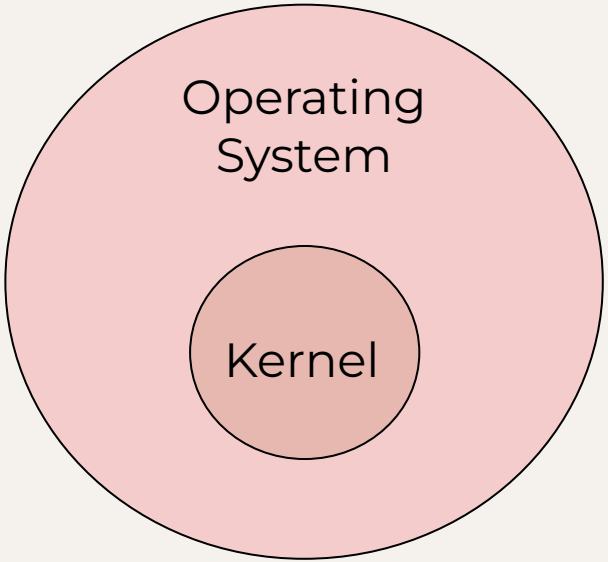
Tutorial 2

Dr. Eng. Catherine M. Elias
Eng. Yasmin El-behiry
Eng. Sarah Khaled
Eng. Farah Shams
Eng. Youssef Saeed

GUC
German University in Cairo

Kernel

- Is the core of the operating system



Kernel

- Responsible for:
  - process management
    - Refers to the activities involved in managing the execution of multiple processes in an operating system.
    - It includes **creating, scheduling, and terminating processes** and many more.

Kernel

- Responsible for:
  - process management
  - memory management
    - Refers to the management of the main memory and disk during process execution.
    - It includes **memory allocation** and many more.

Kernel

- Responsible for:
  - process management
  - memory management
  - file system management
    - Refers to the processes and techniques involved in **creating, organizing, accessing, and controlling files** stored on storage devices.
    - It includes tasks such as **file creation, deletion,** and many more.

Kernel

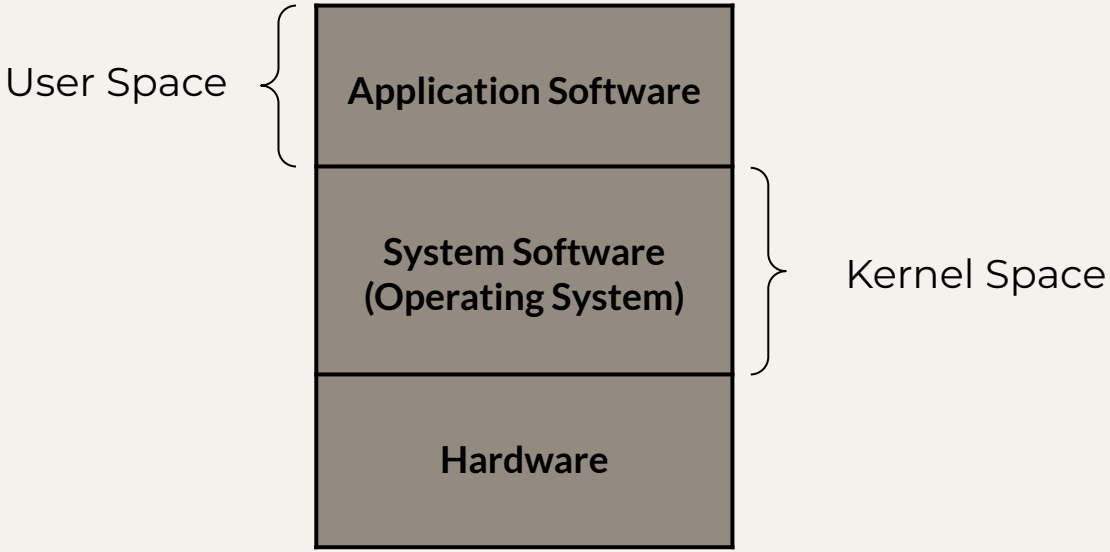
- Responsible for:
  - process management
  - memory management
  - file system management
  - device management
    - Refers to **controlling the Input/Output devices**
    - responsible for managing **device communication** through their **respective drivers**

Kernel

- Responsible for:
  - process management
  - memory management
  - file system management
  - device management
  - system call handling
    - A system call is a **mechanism** used by programs to **request** services from the operating system.

1. What is the difference between Kernel mode and user mode?


RECALL


User Space
Application Software
System Software
(Operating System)
Hardware
Kernel Space

User Mode

- Unprivileged
- Limited access to resources

vs

Kernel Mode

- Privileged
- Full access to resources

2. What is an interrupt?

Interrupts

- Signal emitted by hardware when an event needs immediate attention.
- Examples include but not limited to:
  - moving a mouse
  - pressing a keyboard key

3. Why use Interrupts?

Interrupts

- Interrupts are provided primarily as a way to **improve processor utilization**.
- Most I/O devices are **much slower** than the processor.
- With interrupts, the processor can be engaged in executing other instructions while an I/O operation is in progress.

4. What is an Exception?

Exceptions/Trap

- Events that interrupt the normal execution of a program or a system.
- They can be caused by errors in the software.
- Examples include but not limited to:
  - Null pointer exception
  - Index out of bounds

Interrupt

- Asynchronous
- Hardware

vs

Exception

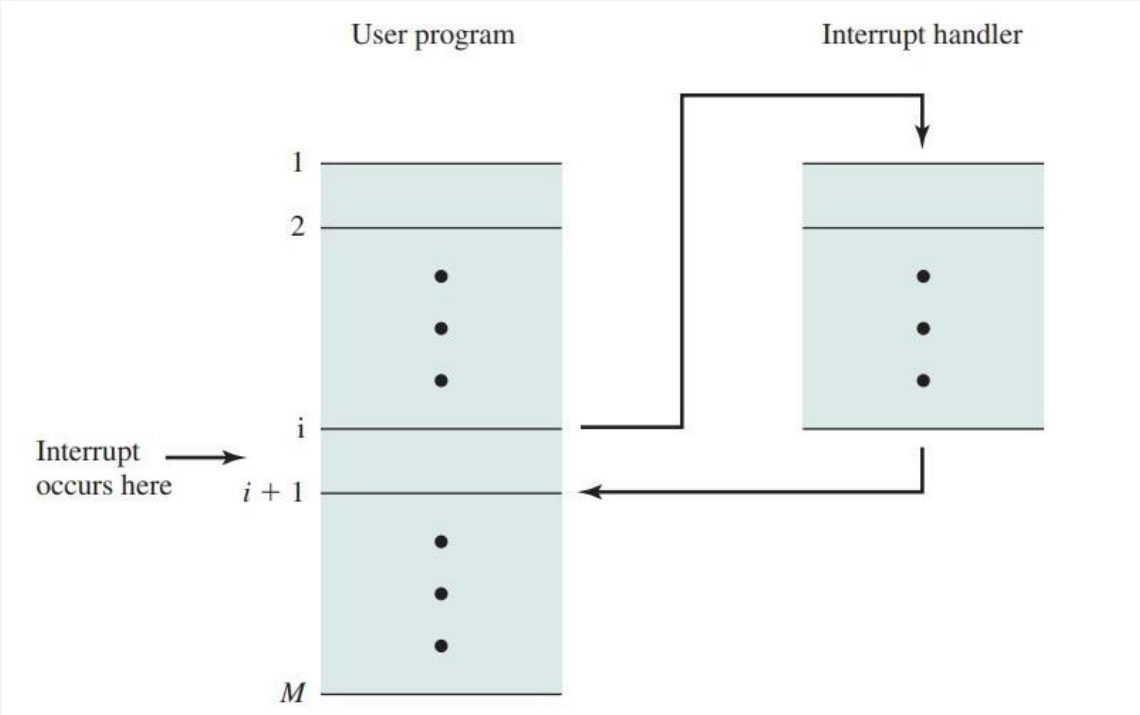
- Synchronous
- Software

5. What happens when an interrupt\Exception occurs?

When An interrupt\Exception Occurs

- Switch from User mode to Kernel mode.
- Stop the currently executing process.
- The appropriate interrupt\exception handler is dispatched.
- After the interrupt is handled, resume execution of the process.

When An interrupt Occurs



6. Imagine what happens when many interrupt requests happen at the same time. What are the different approaches of handling multiple interrupts?

Disabling\Enabling Interrupts

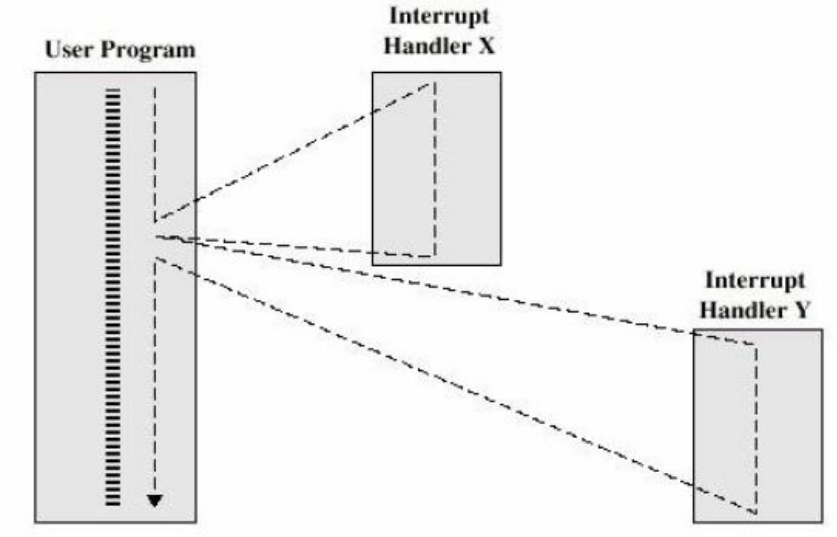


Figure 1: Sequential interrupt processing

Priority Based

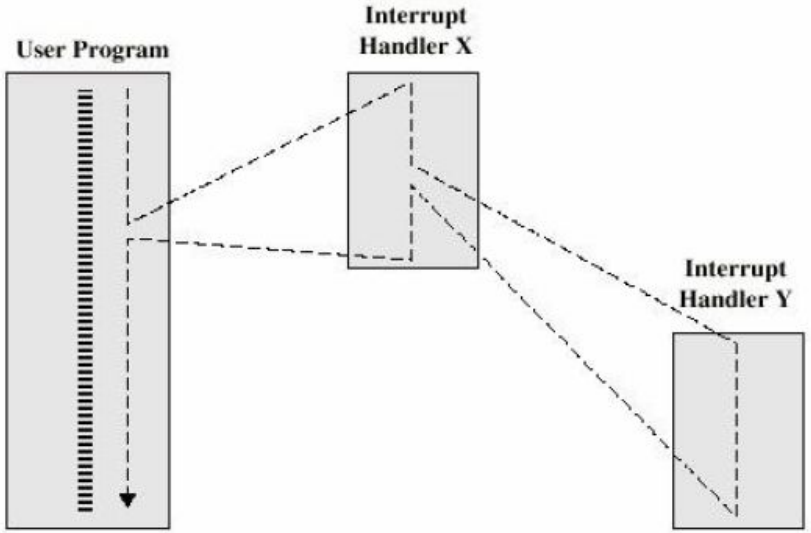


Figure 2: Nested interrupts

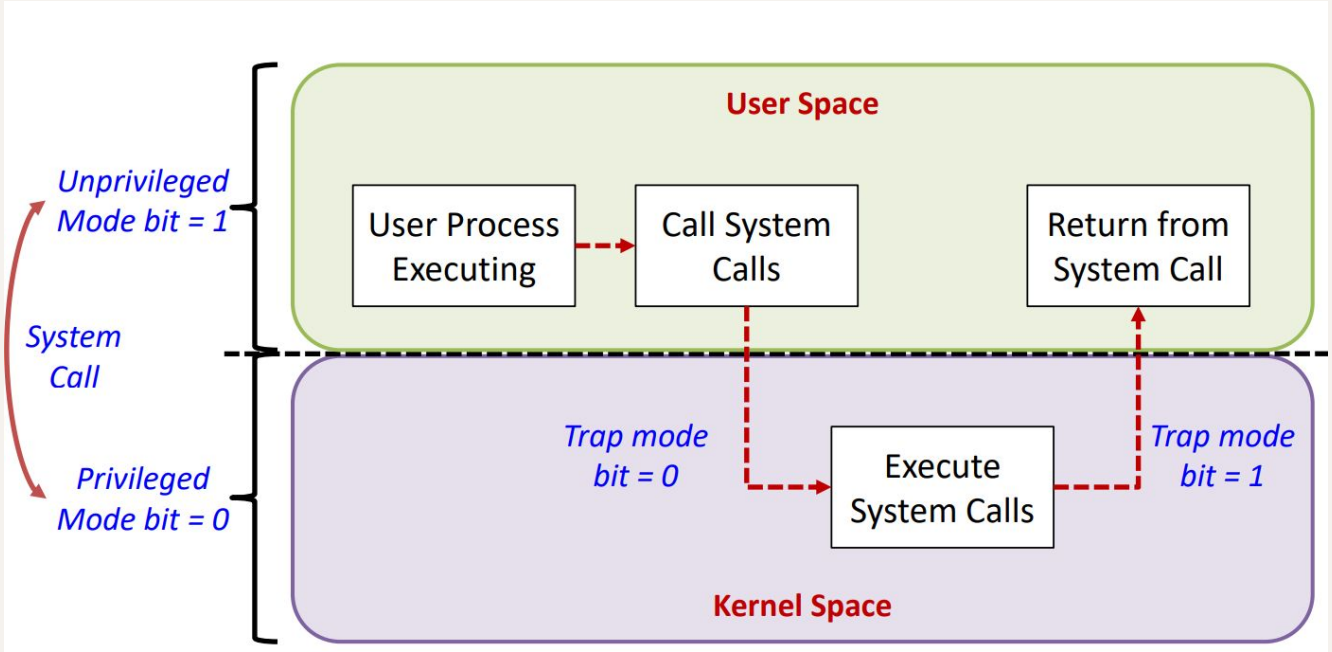
System Calls

- A system call is a **mechanism** used by programs to **request** services from the operating system.
- In simpler terms, it is a way for a program to interact with the underlying system, such as accessing hardware resources.
- Through APIs
- A type of exception

Application programming interface (API)

- By using APIs, applications do not need to know the specifics of the hardware or the OS and can run on different platforms and devices with minimal changes.

System Calls



Examples of System Calls

	Windows	Linux
Process Control	ExitProcess()	Exit()
File manipulation	CreateFile() ReadFile() WriteFile()	Open() Read() Write()
Information Maintenance	GetCurrentProcessID()	Getpid()

All Done!