# COMP-SCI-431
# Intro Operating Systems

## Lecture 1 – Introduction

Adu Baffour, PhD

University of Missouri-Kansas City
Division of Computing, Analytics, and Mathematics
School of Science and Engineering
aabnxq@umkc.edu

UMKC

# Lecture Objectives

- To understand the fundamental role of an operating system in computer systems.
- To describe the structure of an operating system, including its key components and how they interact.
- To illustrate how system calls are used to provide operating system services.
- To examine the evolution of operating systems, exploring their historical development and the current scope of their functionalities.

# Outline

1.1 The role of an operating system

1.2 The OS structure

1.3 The evolution and scope of OSs

UMKC

# Outline

1.1 The role of an operating system

1.2 The OS structure

1.3 The evolution and scope of OSs

# 1.1 The role of an operating system

**Bridging the hardware/user gap**

- The operating system (OS) is the software that runs on the bare hardware of a computer and provides essential support for users to develop and use applications most efficiently and safely.
- The mismatch between hardware capabilities and user needs

| Hardware component | Hardware capabilities | User needs |
|---|---|---|
| CPU | Machine instructions perform operations on contents of registers and memory locations. | The user thinks in terms of arrays, lists, and other high-level data structures, accessed and manipulated by corresponding high-level operations. |
| Main memory | Physical memory is a linear sequence of addressable bytes or words that hold programs and data. | The user must manage a heterogeneous collection of entities of various types and sizes, including source and executable programs, library functions, and dynamically allocated data structures, each accessed by different operations. |
| Secondary storage | Disk and other secondary storage devices are multi-dimensional structures, which require complex sequences of low-level operations to store and access data organized in discrete blocks. | The user needs to access and manipulate programs and data sets of various sizes as individual named entities without any knowledge of the disk organization. |
| I/O devices | I/O devices are operated by reading and writing registers of the device controllers. | The user needs simple, uniform interfaces to access different devices without detailed knowledge of the access and communication protocols. |

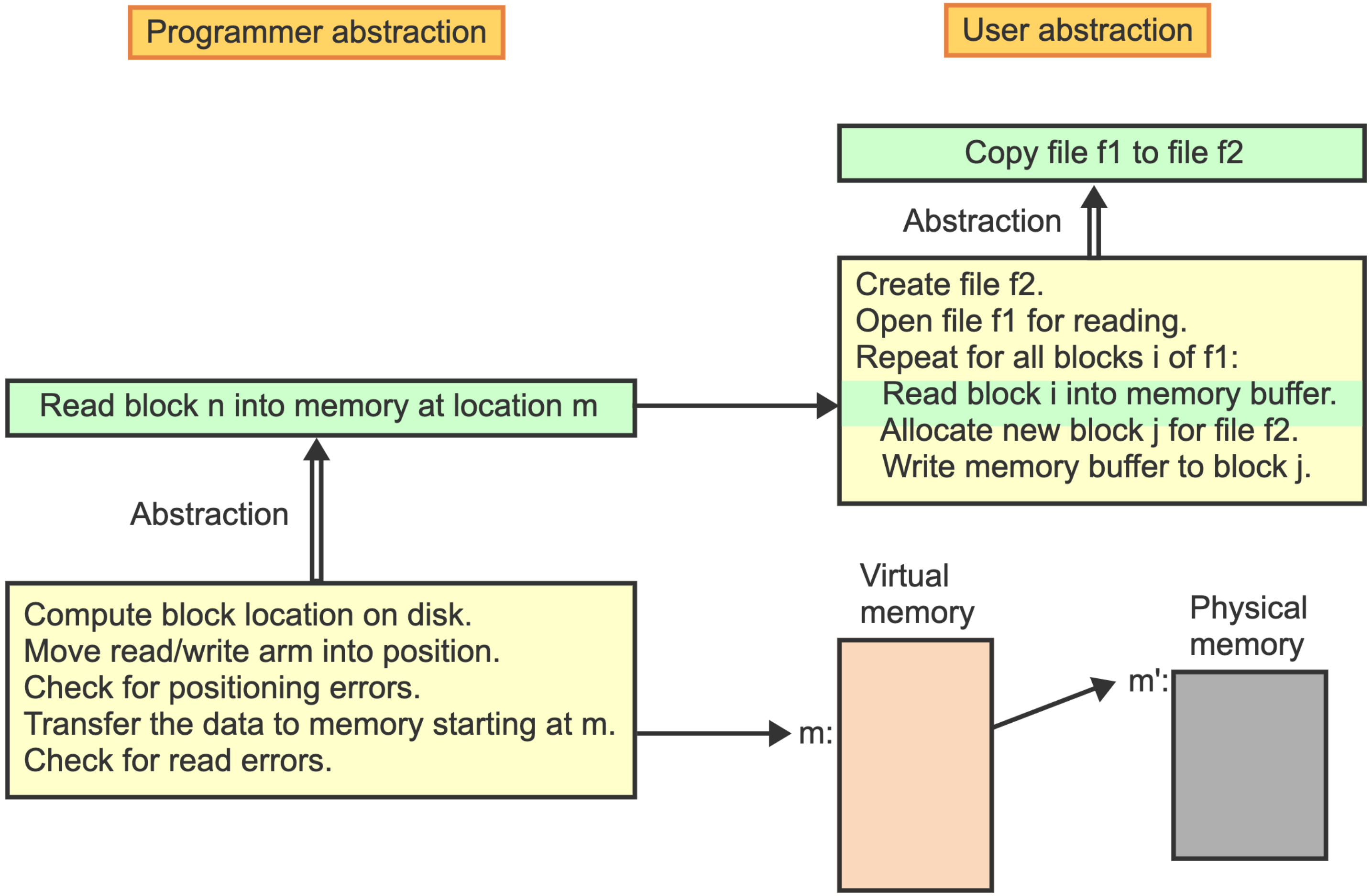
# 1.1 The role of an operating system

**The OS as an extended machine**

- OSs use abstraction extensively by creating hierarchies of objects where multiple operations at one level are combined into a single operation at a higher level.
- Thus hiding the implementation details and making the operation easier to use.
- An OS provides efficient high-level functions and virtual entities that liberate the programmer and the user from understanding details of memory, disk, I/O management, and other internal processes.

# 1.1 The role of an operating system

## Principles of abstraction and virtualization



# 1.1 The role of an operating system

**The OS as a resource manager**

- One of the main tasks of an OS is to optimize the use of all computational resources to ensure good overall performance.
- A program typically alternates between phases of input, computation, and output.
- CPU is underutilized during the I/O phases, while the I/O devices are idle during the compute phase.
- Concurrency - the OS strives to keep the CPU, main memory, and all storage and I/O devices busy by overlapping independent operations whenever possible.

# 1.1 The role of an operating system

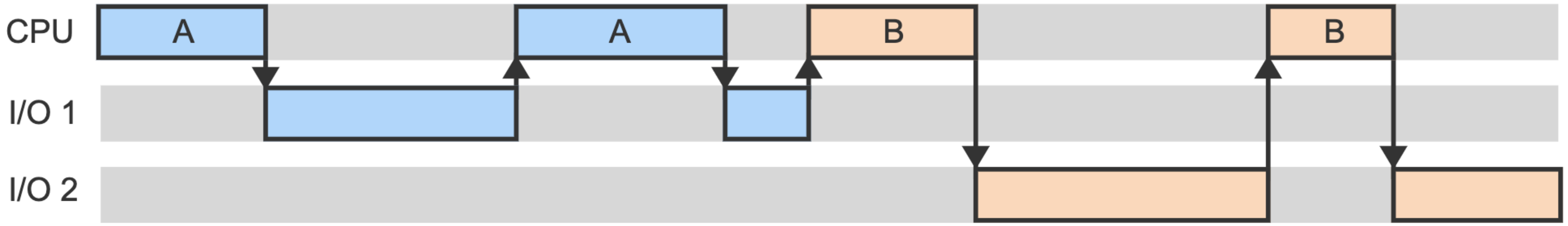
**The OS as a resource manager**

- Multiprogramming is a technique that keeps several programs active in memory and switches execution among the different programs to maximize the use of the CPU and other resources.
- Time-sharing (multitasking) is an extension of multiprogramming where the CPU is switched periodically among all active computations to guarantee acceptable response times to each user.

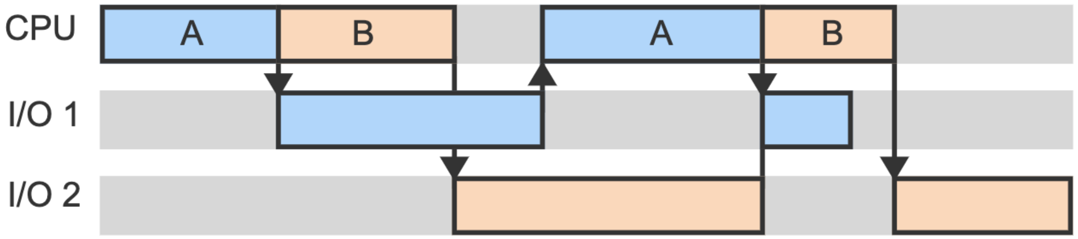
# 1.1 The role of an operating system

## Principles of multiprogramming and time-sharing

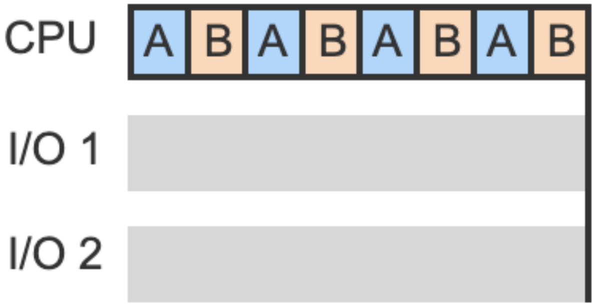
Sequential execution




Multiprogramming




Time-sharing



# Outline

1.1 The role of an operating system

1.2 The OS structure

1.3 The evolution and scope of OSs

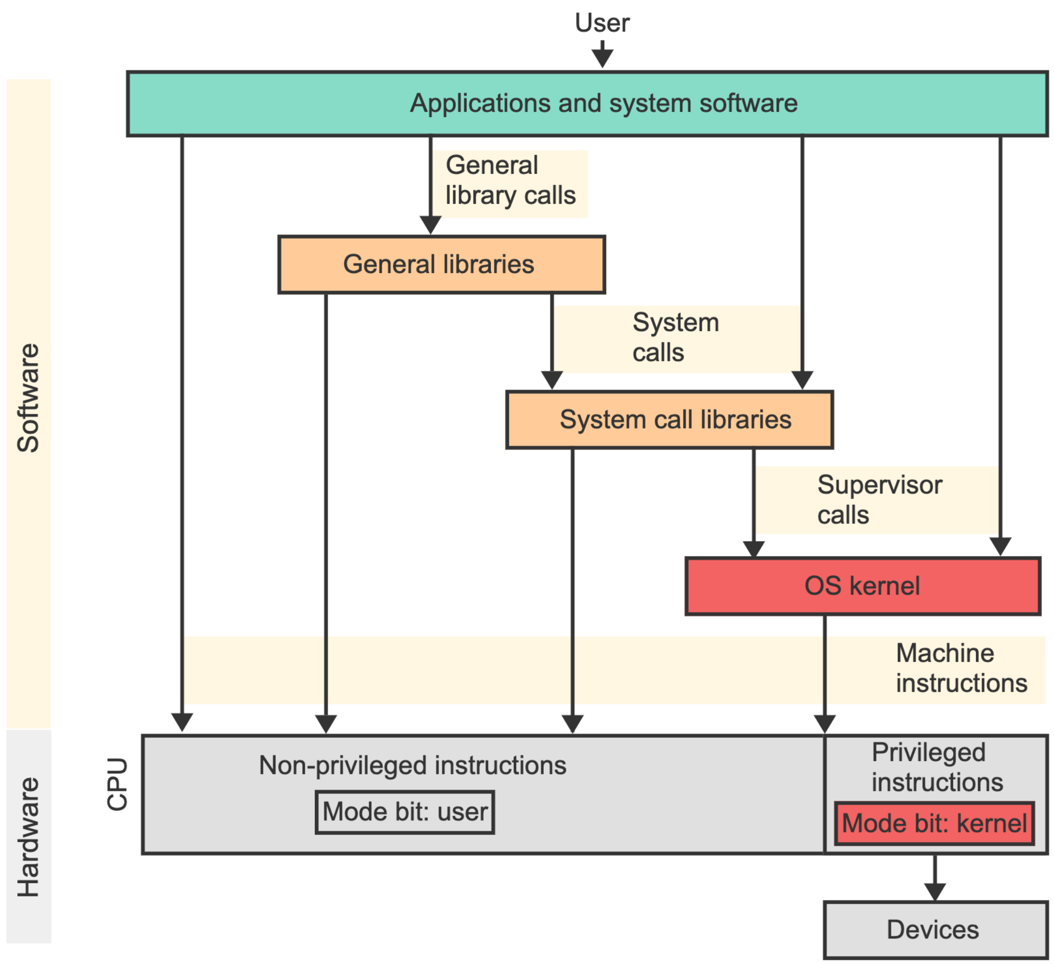
# 1.2 The OS structure

**A hierarchical organization**

- The kernel of an OS is the minimal set of functions necessary to manage the system resources safely and efficiently.
- The CPU's instruction set is divided into privileged and non-privileged to address security issues.
- A privileged instruction performs critical operations that access I/O devices and the CPU's status and control registers. Only the OS kernel is allowed to execute privileged instructions.
- CPU operates in two different modes – kernel mode and user mode, which are indicated by a particular mode bit.
- ***Kernel mode*** is the CPU state where privileged and non-privileged instructions may be used.
- ***User mode*** is the CPU state where only non-privileged instructions may be used.
- Any attempt to execute a privileged instruction in user mode automatically transfers control to the kernel.

# 1.2 The OS structure

The OS hierarchy



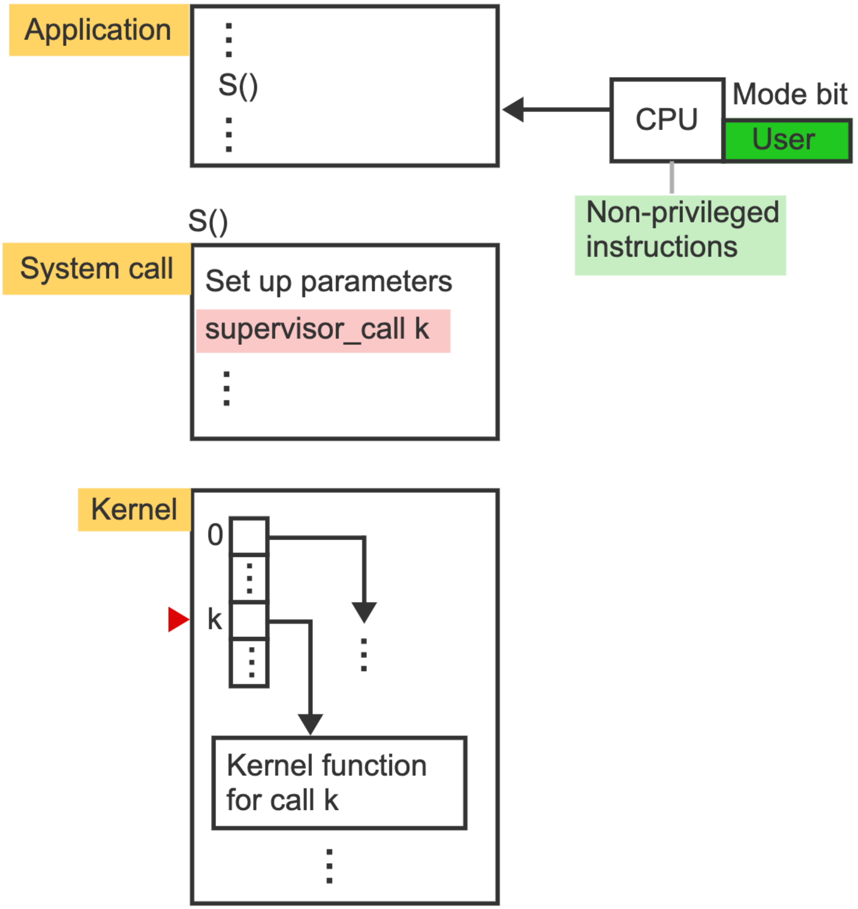
# 1.2 The OS structure

**System calls and supervisor calls.**

- A ***system call*** is a request from an application for an OS service.
- A ***supervisor call*** (***kernel call***) is a privileged instruction that automatically transfers execution control to a well-defined location within the OS kernel.
- A supervisor call is like a function call with two special features:
  - The call switches execution from user to kernel mode by setting the mode bit in the CPU.
  - Matches the function to be invoked using an index into a branch vector. Thus, kernel-mode execution is limited to only well-defined entry points within the kernel.
- When the kernel function terminates, control is returned to the invoking library function in user mode.

# 1.2 The OS structure

Execution of a system call



# 1.2 The OS structure

**Interrupts and traps**

- An ***interrupt*** is an event triggered by an external device's hardware signal that diverts the current execution of a program to a predefined location in the kernel to respond to an event.
- The two most common uses of interrupts are as follows:
  - Signal to the OS the completion of an I/O operation. The I/O device generates the interrupt.
  - Implement time-sharing. A countdown timer generates the interrupt.
- A ***trap*** (an internal interrupt) is triggered by the currently executing instruction. E.g., Dividing by zero.
- Executing a supervisor call instruction is not an error but causes a trap since the primary purpose is to transfer control to the kernel when requesting a service.
- An ***interrupt handler*** is a kernel function invoked whenever an interrupt occurs that determines the cause of the interrupt and invokes the appropriate kernel function to respond.

# 1.2 The OS structure

**Interface**

- The OS starts a graphical user interface or a shell when a user logs in.
- A ***graphical user interface*** (***GUI***) presets various icons on the screen, which the user can click on in different ways to invoke services associated with the icons or to reveal pull-down menus for additional tasks.
- The ***OS shell*** is a command interpreter that accepts and interprets textual commands issued by the user via a keyboard.

# Outline

# 1.3 The evolution and scope of OSs

**The five generations of computer systems**

- Moore's law, formulated by the scientist Gordon Moore, is the observation that the number of transistors in an integrated circuit doubles about every two years.
- 1st generation: Vacuum tubes with no OS
- 2nd generation: Transistors replaced vacuum tubes as smaller and faster switches. Batch OS
- 3rd generation: Integrated circuits allowed the development of microchips to replace individual transistors—interactive multi-user OS.
- 4th generation: Very-large-scale integration (VLSI) allowed the placement of a complete microprocessor on a single chip, leading to the development of personal computers (PCs) —desktop and laptop OS.
- 5th generation: Networking hardware enabled the harnessing of the power of multiple computers —OSs for supercomputers, distributed systems, and mobile devices.

UMKC

# 1.3 The evolution and scope of OSs

## Types of OSs

| Type of operating environment | Common application areas | Major emphases |
|---|---|---|
| Mainframe: a large central computer used by large organizations. | High-volume data processing in administration, banking, government. | High throughput, management of large storage. |
| Server: a large computer that responds to requests from individual clients. | Web and email processing, Internet commerce. | Fast response, security. |
| Multiprocessor: a system of multiple CPUs and memories interconnected by a fast network into a single parallel computer. | Scientific and other high-performance computations. | Fast interprocess communication and memory access. Data consistency. |
| Distributed system (multi-computer): a network of independent computers interconnected via a communication network. | Sharing of data and services, internet commerce. | Efficient and secure communication. Support for many types of applications and services. |
| Desktop or laptop: a personal computer. | Word processing, personal finance, access to Internet, games. | User-friendly interface. Intuitive organization of data and applications. Support for a variety of tasks without much technical knowledge of the inner functioning of the computer. |

UMKC

# 1.3 The evolution and scope of OSs

## Types of OSs

| | | |
|---|---|---|
| Hand-held device: small, portable, wireless-capable device for personal use. | Smartphones, tablets. | User-friendly interface. Easy integration of new applications. Support for microphone, speaker, camera, GPS, motion sensor, and other components. |
| Real-time system: a computer, frequently embedded in a larger electro-mechanical system, that must respond rapidly to external events. | Control of industrial processes, vehicle and aircraft control, audio and video transmission. | Scheduling to meet all deadlines. Reliability in life-critical applications. |
| Sensor network: collection of small, spatially distributed dedicated sensors communicating by wired or wireless connections. | Industrial, environmental, or military monitors. Wearable devices. | Minimize power consumption. Form ad-hoc connections and tolerate node failures. |

# End of Lecture

Thank you

Any questions?