1 What is an Operating System?

An operating system is the software that manages the computer hardware. It is composed of a collection of specific-purpose programs that run on a computer system and allow it to function properly. These programs perform basic tasks such as recognizing keyboard and mouse inputs, keeping track of stored files and folders, sending information to the display, or controlling peripheral devices.

The purpose of an operating system is to provide an environment in which users can run programs conveniently and efficiently. It is composed of essential programs that manage the computer hardware and provide a user-friendly interface for running applications. These programs and software modules serve as an intermediary between the hardware and the applications running in a computer system.

Therefore, operating systems have several tasks:

- They manage system resources, such as the processor, memory, and input and output devices, allocating these resources to applications, ensuring that they do not interfere with each other.
- They abstract away the computer hardware, preventing users and programmers from having to know its details and providing a simple and coherent interface to run applications, hiding the complexity of the underlying hardware.
- They provide security and protection to applications, offering mechanisms to protect system resources and data from unauthorized access and malicious software.
- They provide utilities and services that make it easier for users to interact with the computing system and run applications.

System calls are symbolic constructs that provide a way for user programs to request services and resources from the underlying operating system. In other words, system calls act as a bridge between user space and operating system space (cf. Fig. 1). When a user program needs to perform privileged operations or access system resources, it makes a request to the operating system through a system call. The operating system, which operates in a more privileged mode, receives and handles these requests, performing the necessary operations on behalf of the user program.

Some common examples of system calls include those to perform file operations (e.g., opening, reading, writing, closing files, and manipulating file attributes); to control processes (e.g., creating, terminating, and managing processes); to manage memory (e.g., allocating and freeing memory, mapping and unmapping memory areas); to interact with input/output devices (e.g., performing operations such as reading from or writing to a device); to communicate through a network (e.g., establishing network connections, sending and receiving data over networks); or to exchange information among processes (e.g., synchronizing and communicating among different processes).

System calls provide an abstraction layer for user programs, allowing them to access system resources and services in a controlled and secure manner. They provide a standardized interface for applications to interact with the underlying operating system, hiding the complexities of low-level operations and providing a higher-level programming interface.

Operating systems can be classified into several types, such as batch processing systems, time-sharing systems, distributed systems, network operating systems, real-time operating systems and mobile operating systems. Each type of operating system is designed for an