Luiza Yeghiazaryan Essay - OS architecture

An operating system is not just a program, but a complete system that manages how the computer works. It acts as a bridge between the user, applications, and hardware.

An operating system is usually described in layers. At the top is the UI (User Interface) which lets us interact with the system. Below are system calls and APIs (Application Programming Interfaces). System calls are low-level instructions the the kernel, while APIs let programmers use them without handling hardware directly. The kernel is the part of the OS that directly interacts with the hardware. There are also layers between the user and the kernel that organize the computer, making it safe and easy to use. These include the shell (lets users give commands), system utilities (run in the background to manage files or networks), and security (controls access and protects files).

The kernel is the most important part of the operating system. It performs several critical functions. It ensures fair CPU time for all programs, allocates separate memory for each program to prevent conflicts, organizes files on disk and communicates with hardware through device drivers. Without the kernel, applications could conflict, memory could be corrupted and hardware would not function properly.

There are three main types of kernel.

Monolithic kernels handle almost all services, including device drivers, file management, and memory - in one big code. This makes them fast, but if one part fails, it can affect the entire system.
Microkernels keep only essential functions in the kernel, while other services run separately in user space. This structure is safer and easier to maintain, but slightly slower.
Hybrid kernels combine both approaches, grouping some tasks in the kernel for speed and running others separately for safety and modularity. Most modern operating systems like Windows and MacOS, use hybrid kernels, because they balance efficiency and reliability.
By studing these three types, it is clear why kernel design affects both speed and reliability, and why engineers choose different approaches depending on the system's goals.
One of the most important lessons I learned is how layers interact: User space → System calls → kernel → hardware

User programs request services via APIs, which ~~needs~~ send systems calls to the kernel, which checks permissions, gets data from hardware and returns it to the program. Each layer does its own job, keeping the system smooth, safe and organized.

This simple model shows why OS architecture is crucial