Evolution of Operating Systems

The evolution of operating systems has been driven by the need to keep pace with improvements in hardware and to offer new services, such as internet support. This requires the design of operating systems to be modular, with clean interfaces and an object-oriented methodology.

The timeline of operating system evolution is broken down into several stages:

Early Systems (1950s): These were single-user systems where a programmer acted as the operator, running large machines from a console using punched cards or paper tape. This era, also known as Serial Processing, was inefficient due to slow I/O devices, significant setup time, and very low CPU utilization. Only one program could be executed at a time, and there was no user-to-computer interaction.

Simple Batch Systems (1960s): To improve efficiency, jobs were batched together by type, and an operator was hired to manage the process. A special program called the monitor managed the execution of jobs in the batch. The Job Control Language (JCL) was introduced to provide instructions to the monitor, such as which compiler to use and what data was needed. The "resident monitor" was a rudimentary operating system that automatically transferred control from one job to the next, reducing idle time between programs.

Multiprogrammed Batch Systems (1970s): Multiprogramming allows multiple processes to reside in main memory at the same time, and the CPU is multiplexed among them. When one program is waiting for an I/O operation, the CPU can be allocated to another program, keeping the CPU busy and increasing its utilization. This led to two types of multiprogramming operating systems:

Multitasking OS: Capable of executing many programs at once by swapping them in and out of memory. Examples include Windows XP, Vista, and Windows 7.

Multiuser OS: Allows multiple users to connect to a single system running the same operating system. Examples include Windows 2000, Ubuntu, and Mac OS.

Time-Sharing and Real-Time Systems (1970s): A logical extension of multiprogramming, time-sharing systems allow several jobs to be loaded into main memory simultaneously, with each process sharing an equal amount of CPU time. This improved response time and CPU utilization.

Real-time systems are designed for applications where the correct system function depends on timeliness, with categories like hard real-time (where failure occurs if the response time is too long) and soft real-time (where accuracy decreases if the response time is too long).

Personal/Desktop Computers (1980s): These are portable, single-user systems with devices like keyboards, mice, and display screens. They prioritize user convenience and responsiveness.

Multiprocessor Systems (1980s): These systems feature more than one CPU in close communication, leading to improved throughput, reliability, and cost-effectiveness. A single job can be divided into smaller tasks and executed in parallel across multiple processors.

Networked/Distributed Systems (1980s): These systems distribute computation across many processors that are "loosely coupled" with no shared memory, communicating via various lines. Advantages include resource sharing, increased computation speed, and improved reliability. Types of distributed systems include Client-Server Systems, which provide a centralized server for requests, and Peer-to-Peer Systems, where processors have their own local memory and communicate with each other.