Operating Systems

An operating system is the underlying software that controls the systems resources and how programs are run executed.

Types of Interface

An operating system must provide multiple types of interfaces.

The operating system is used by different types of users for different purposes. As such different interfaces are provided to enable the various use cases.

GUI

The graphical user interface is user-friendly and allows non-technical users to leverage computers. Early computers did not have graphical user interfaces.

Command-line

The command line interface is a textual interface. Command line interfaces can allow developers to string together command line programs and so can offer greater flexibility for developers than a GUI.

API

Operating system expose a programming interface that allows developers to pragmatically control system resources such as files, this allows developers to create richer software.

Hardware interfaces - device drivers

The device divers enable the hardware to interact with the operating system.

Resource management and scheduling

The operating system manages CPU, memory, file, process management, energy management and communications.

A process is the memory image of a program. This image contains the executable code as well as 'live' aspects such as variable values (on the stack) and the current executing line number. Processes may utilize multiple threads (Ie separate threads for GUI and communication). The process is the unit that is allocated for a program to execute and the thread is the unit a program allocates for its execution.

The operating system provides a virtualizing of the hardware allowing developers to write hardware agnostic code. On Unix systems all hardware's drivers are modeled as files so the same common file interface can be used to access any piece of hardware.

Types of operating system kernels

There are three primary types of operating system kernels; monolithic, micro and exokernels. A monolithic kernel includes all services that are required for operation. Micro and exokernels however allow modular pieces of the kernel to be loaded on demand. A micro kernel has some services whereas an exokernel has no features (only hardware virtualization) and all modules must be loaded on demand.

The operating system provides security for the users and applications it runs.

Process Scheduling

Each process is allocated time slices by the operating system. There are various ways that the operating system can decide which process gets served next. A simple solution is called a round robin. In a round robin each is served in a loop with no preference. Alternatives include priority's and first come first served ques. Scheduling on multicores is more complicated a can lead to concurrency and deadlock issues.

Operating system history

Timesharing systems where invented in 1959 and laid the ground work for modern systems that allow multiple users to use them at once.

The first computers each had their own unique operating systems, this meant software was not compatible across machines. The IBM 360 was the first collection to share the same operating system across different computers.

The first operating system for the 'microcomputer' was the Control Program/Monitor. The CP/M separated hardware drivers and system software. The hardware drivers are hardware specific and form the BIOS component.

The first OS with a graphical interface was demonstrated in in 1968, it wasn't until the 1980s however that GUIs became widely used.

Mobile operating systems. In the early days most mobile phones ran on a Symbian operating system. Later Android and IOS became the dominate forces. Android is open source and can run on many different phones and in many different versions(flavors). IOS comes in a smaller number of discrete versions and runs on a limited set of hardware(iPhone and tablets).

Jerome Saltzer and Michael Schroeder

In 1975 Jerome Saltzer and Michael Schroeder wrote an influential article called The Protection of Information in Computer Systems. In this article they describe ten design principles for achieving protection.

The key principles that I think are still applicable today are:

- Economy of Mechanism
- Keep you solution as simple as possible.
- Fail-safe Defaults
- By default, deny all access, then only allow access to what is trusted.
- Separation of Privilege

Use multiple checks before granting permission, this is similar to multi-factor authentication.

Least Privilege

Only grant the minimum permissions necessary for an application to complete its job. 'sandbox' environments can be used to isolate aspects.

Secure programming languages

Secure software should be written in secure programming languages. Modern languages such as Go and Rust are designed to be memory and thread safe where C was not. Insecure languages have vulnerability such as buffer overflows.

Secure Protocols

Secure protocols such as HTTPS should be used for communications. Secure protocols ensure confidentiality, integrity and availability.

Applications should have 'App Armor'. App Armour should limit the security risks of applications on the operating system it runs on.

As well as hardware virtualization discussed earlier. Operating system virtualization can also be achieved. A virtual operating system runs on an existing OS and any application on the virtual OS cannot affect the original OS.

There are legal, ethical, social and profession consideration involved in the development of operating systems.