Sairam INSTITUTIONS

Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

YEAR	SEM
II	IV

CS8493

OPERATING SYSTEMS

UNIT No. 1

1.7 Operating System Structures and Operations

Version: 1.XX

Operating-System Structure

An operating system is a construct that allows the user application programs to interact with the system hardware. For efficient performance and implementation an OS should be partitioned into separate subsystems, each with carefully defined tasks, inputs, outputs, and performance characteristics.

Operating systems can be implemented with the help of various structures. The structure of the OS depends mainly on how the various common components of the operating system are interconnected and melded into the kernel.

Depending on this we have following structures of the operating system:

- Simple structure – MS-DOS
- More complex -- UNIX
- Layered – an abstraction
- Microkernel -Mach
- Modules
- Hybrid Systems

- **Simple Structure**
    - A common example of this is MS-DOS.
    - MS-DOS – written to provide the most functionality in the least space.
    - Not divided into modules
    - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated
    - It does not break the system into subsystems, and has no distinction between user and kernel modes, allowing all programs direct access to the underlying hardware.

An image to illustrate the structure of MS-DOS is as follows −

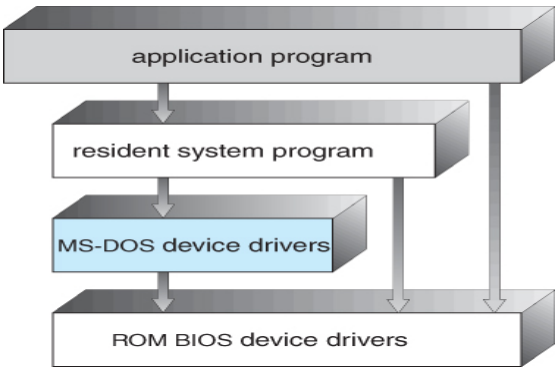
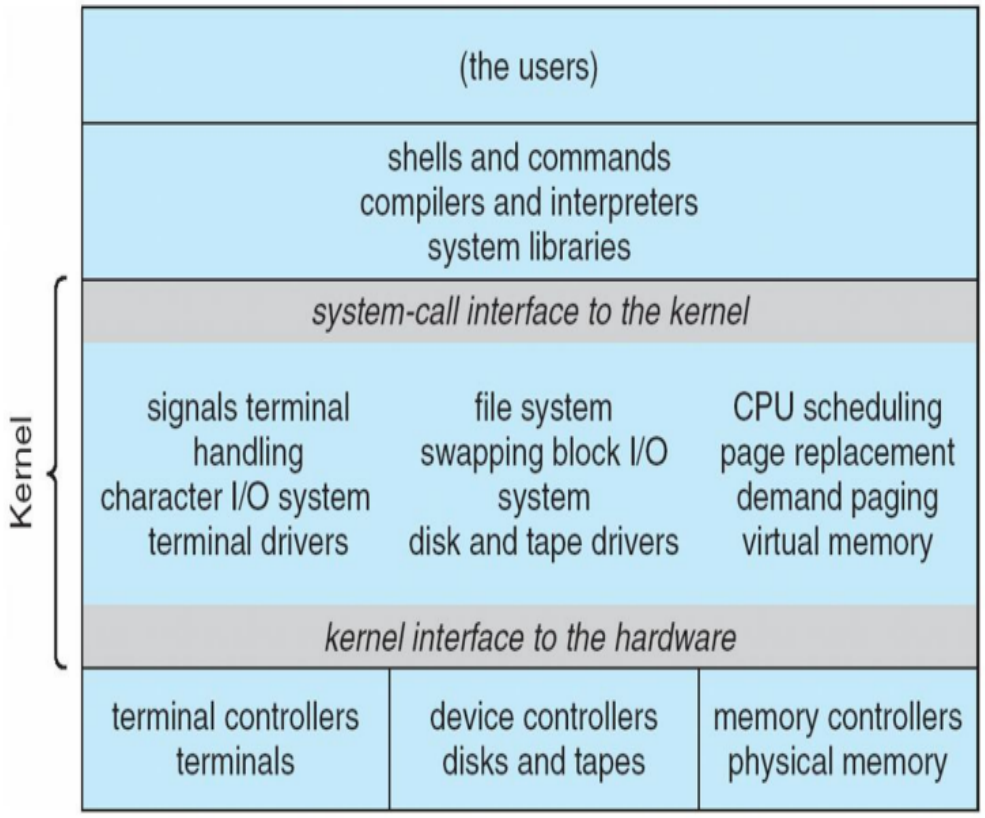


Figure 1 - MS-DOS layer structure

- **Non Simple Structure -COMPLEX**
  - A common example of this is UNIX
  - UNIX – limited by hardware functionality, the original UNIX operating system had limited structuring.
  - The UNIX OS consists of two separable parts
    - **Systems programs**
    - **The kernel**
      - Consists of everything below the system-call interface and above the physical hardware
      - Provides the file system, CPU scheduling, memory management, and other operating-system functions;a large number of functions for one level

Traditional UNIX System Structure



- **Layered Structure**

One way to achieve modularity in the operating system is the layered approach.This approach breaks up the operating system into different layers.

- This allows implementers to change the inner workings, and increases modularity.
- As long as the external interface of the routines don't change, developers have more freedom to change the inner workings of the routines.
- With the layered approach, the bottom layer is the hardware, while the highest layer is the user interface.
    - The main *advantage* is simplicity of construction and debugging.
    - The main *difficulty* is defining the various layers.
    - The main *disadvantage* is that the OS tends to be less efficient than other implementations.

An image demonstrating the layered approach is as follows −

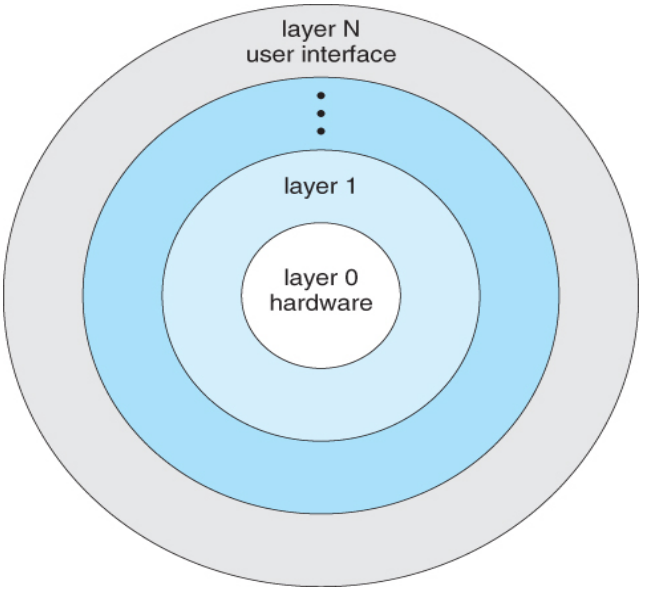


Figure 2 - A layered operating system

- **Microkernels**

This structures the operating system by removing all nonessential portions of the kernel and implementing them as system and user level programs.

- Generally they provide minimal process and memory management, and a communications facility.
- Communication between components of the OS is provided by message passing.

The *benefits* of the microkernel are as follows:

- Extending the operating system becomes much easier.
- Any changes to the kernel tend to be fewer, since the kernel is smaller.
- The microkernel also provides more security and reliability.

Main *disadvantage* is poor performance due to increased system overhead from message passing.

- Another microkernel example is QNX, a real-time OS for embedded systems.

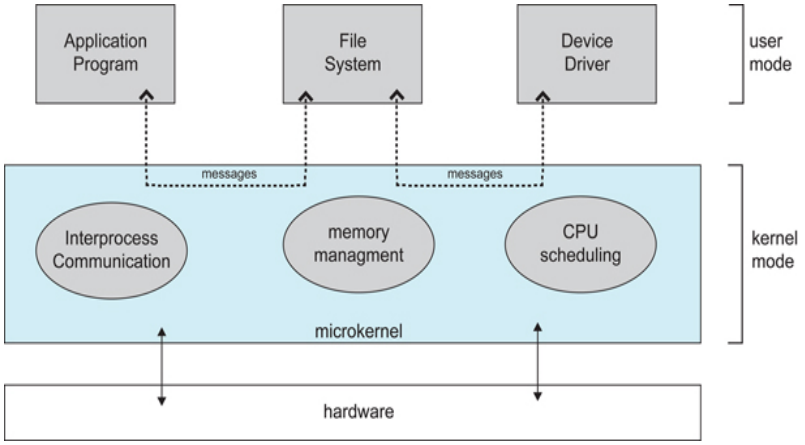


Figure 3 - Architecture of a typical microkernel

- **Modules**
  - Modern OS development is object-oriented, with a relatively small core kernel and a set of ***modules*** which can be linked in dynamically. See for example the Solaris structure, as shown in Figure 4 below.
  - Modules are similar to layers in that each subsystem has clearly defined tasks and interfaces, but any module is free to contact any other module, eliminating the problems of going through multiple intermediary layers, as well as the chicken-and-egg problems.
  - The kernel is relatively small in this architecture, similar to microkernels, but the kernel does not have to implement message passing since modules are free to contact each other directly.

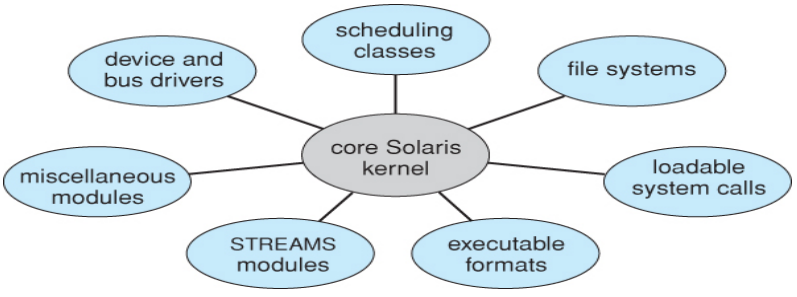


Figure 4 - Solaris loadable modules

- **Hybrid Systems**

Most OSes today do not strictly adhere to one architecture, but are hybrids of several.

- **Mac OS X**

The Mac OSX architecture relies on the Mach microkernel for basic system management services, and the BSD kernel for additional services. Application services and dynamically loadable modules (kernel extensions) provide the rest of the OS functionality:

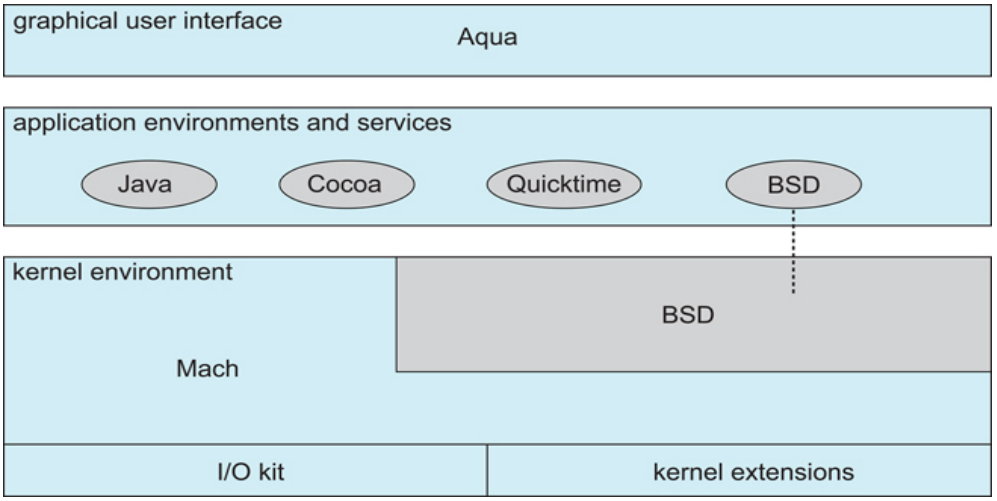


Figure 5 - The Mac OS X structure

- **iOS**

The iOS operating system was developed by Apple for iPhones and iPads. It runs with less memory and computing power needs than Max OS X, and supports touchscreen interface and graphics for small screens:

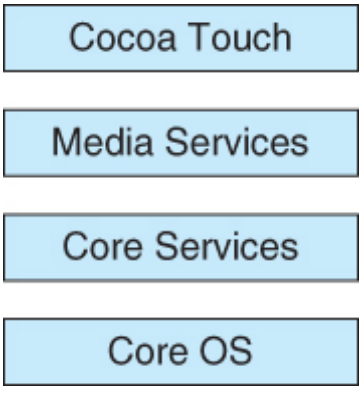


Figure 6 - Architecture of Apple's iOS

- **Android**
  - The Android OS was developed for Android smartphones and tablets by the Open Handset Alliance, primarily Google.
  - Android is an open-source OS, as opposed to iOS, which has lead to its popularity.
  - Android includes versions of Linux and a Java virtual machine both optimized for small platforms.
  - Android apps are developed using a special Java-for-Android development environment.

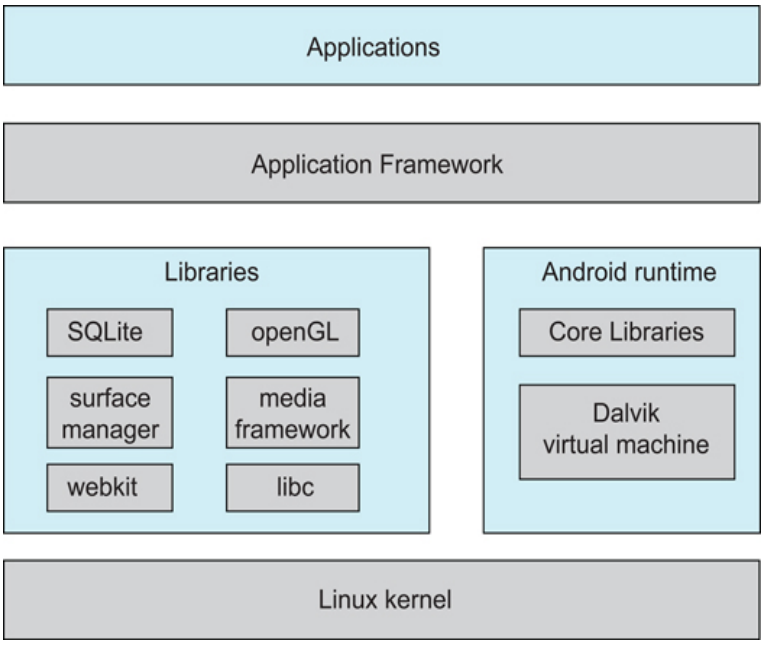


Figure 7 - Architecture of Google's Android