ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Operating Systems

Lecture 2: Operating Structures

© Fall 2024 – **Dr. Alyaa Hamza**
Alyaa.hamza@sut.edu.eg

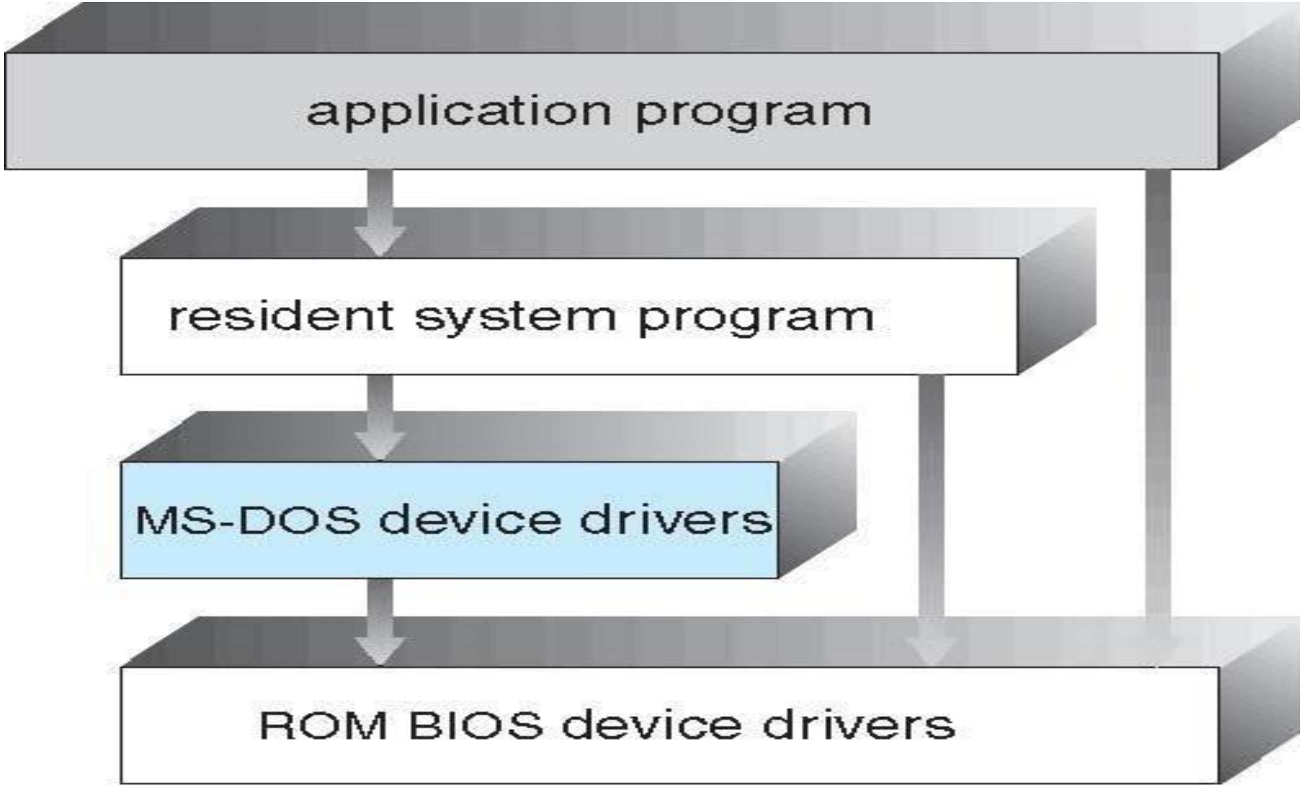
Operating-System Structure

- A system as large and complex as a modern operating system must be engineered carefully if it is to function properly and be modified easily.
- A common approach is to partition the task into small components, or modules, rather than have one **monolithic** system.
- Each of these modules should be a well-defined portion of the system, with carefully defined inputs, outputs, and functions.

Simple Structure

- Many operating systems do not have well-defined structures. Frequently, such systems started as small, simple, and limited systems and then grew beyond their original scope.
- MS-DOS is an example of such a system.
- 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. For instance, application programs are able to access the basic I/O routines to write directly to the display and disk drives.
  - Such freedom leaves MS-DOS vulnerable to errant programs, causing entire system crashes when user programs.

MS-DOS layer structure

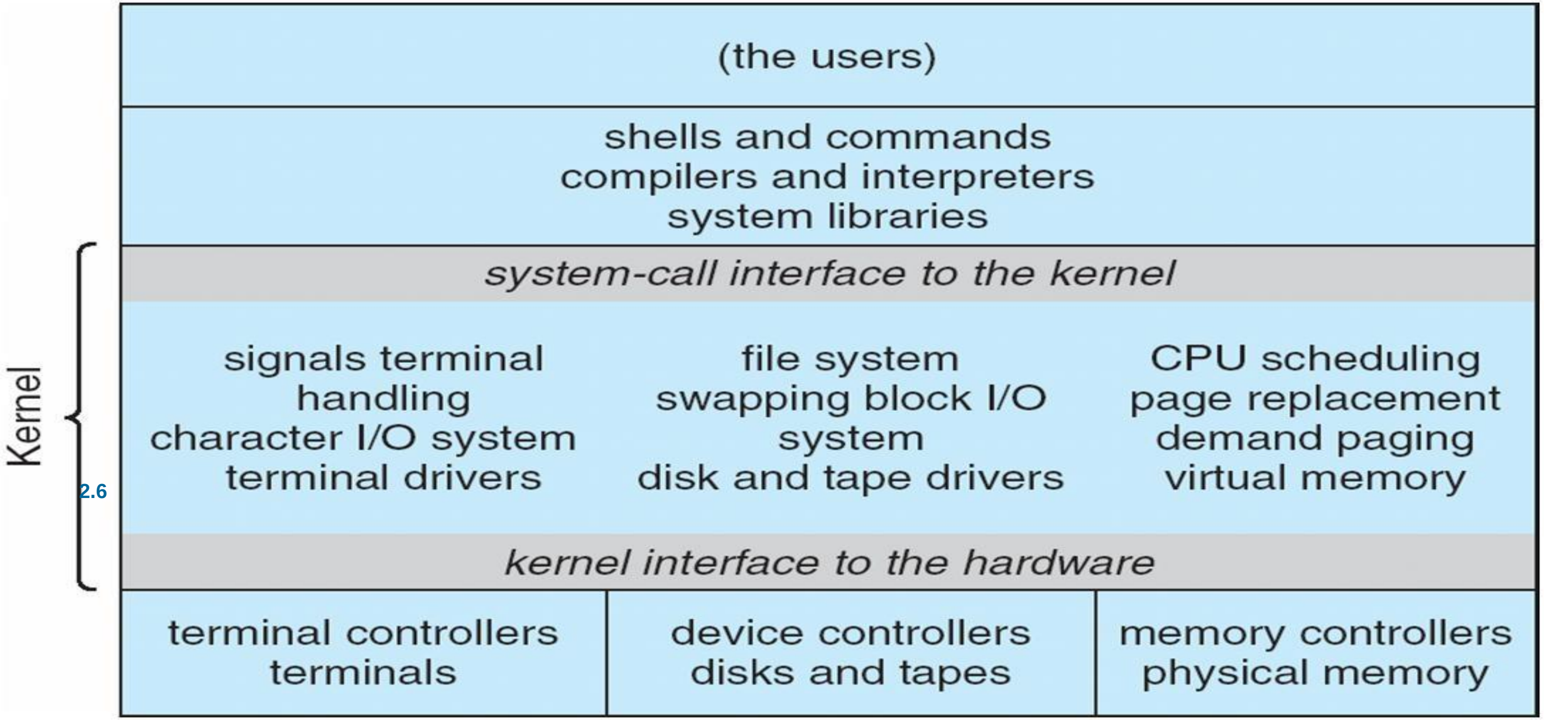


Traditional UNIX

- 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
- This monolithic structure was difficult to implement and maintain.
- It had a distinct performance advantage, there is very little overhead in the system call interface or in communication within the kernel

Traditional UNIX System Structure

Beyond simple but not fully layered

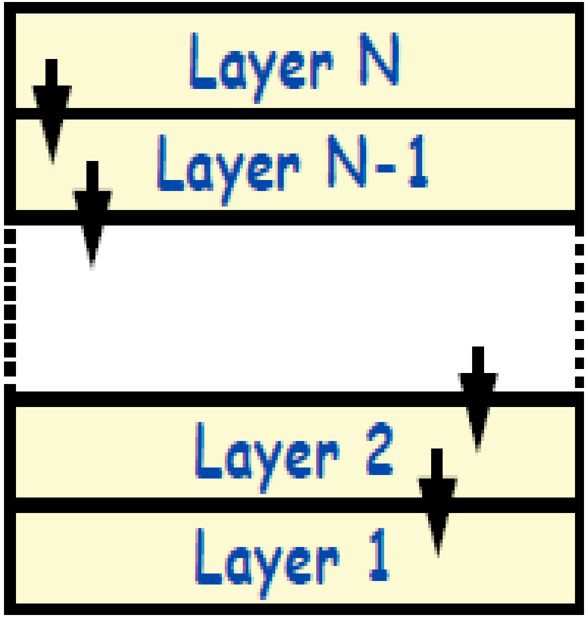


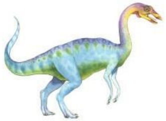
Layered Approach

- The operating system is divided into a number of layers (levels), each built on top of lower layers. The bottom layer (layer 0), is the hardware; the highest (layer N) is the user interface.
- With modularity, layers are selected such that each uses functions (operations) and services of only lower-level layers.
- The main advantage of the layered approach is simplicity of construction and debugging.
- Each layer is implemented only with operations provided by lower-level layers. A layer does not need to know how these operations are implemented.
- The major difficulty involves appropriately defining the various layers.
- A final problem is that they tend to be less efficient than other types

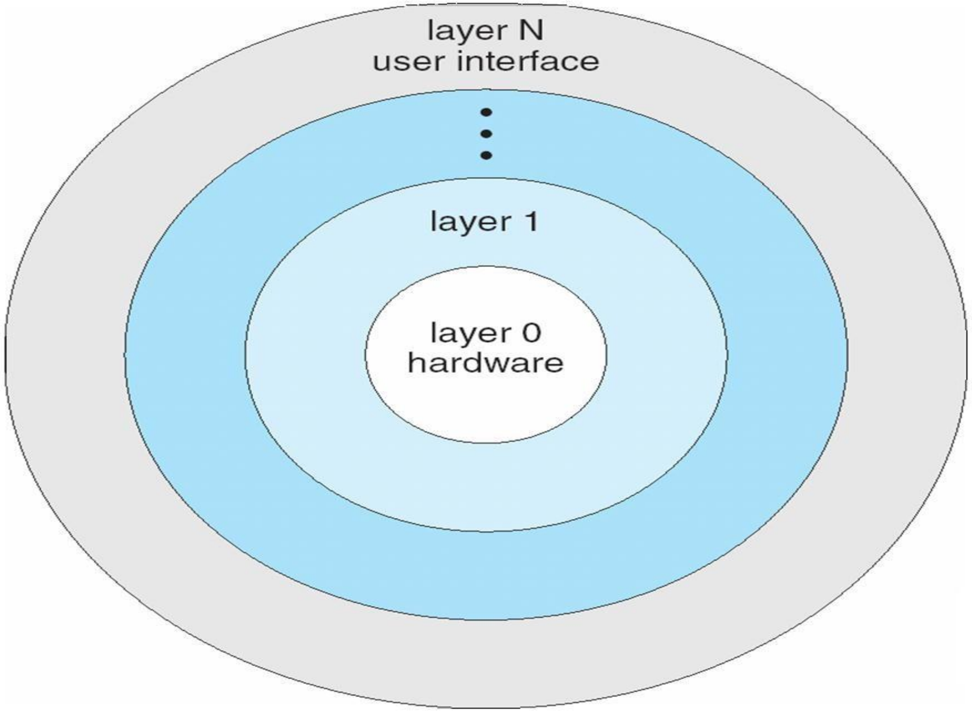
Layered architecture

- Each layer only uses services of the layer immediately below;
- Minimizes dependencies between layers and reduces the impact of a change.

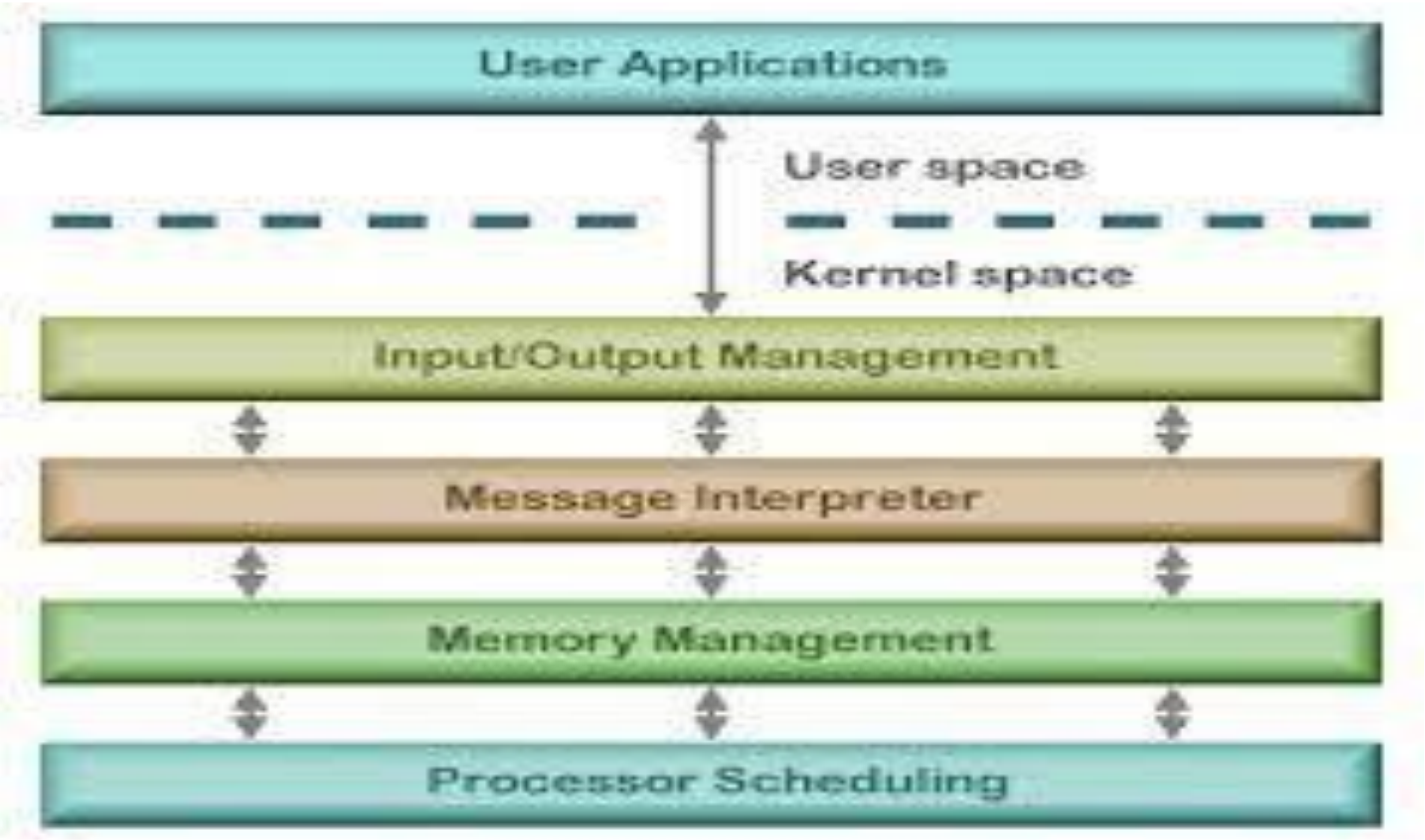




A layered operating system



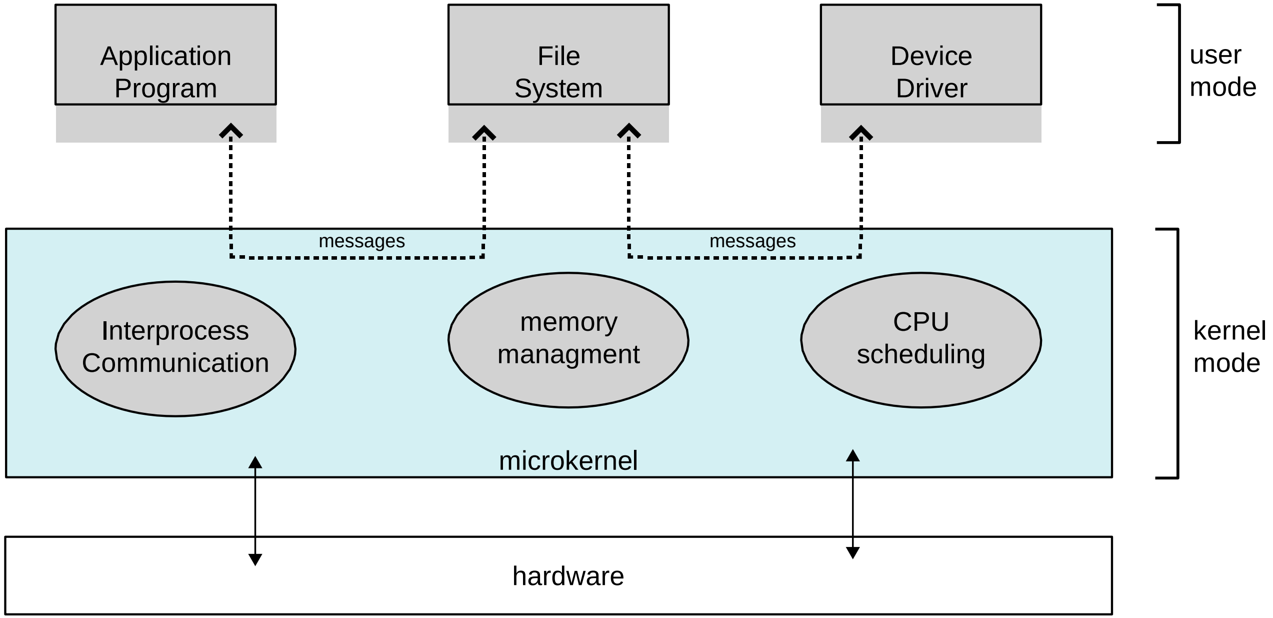
A layered operating system



Microkernel System Structure

- Moves as much from the kernel into user space
- **Mac** example of **microkernel**
  - Mac OS X kernel (**Darwin**) partly based on Mach
- Communication takes place between user modules using **message passing**
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Detriments:
  - Performance overhead of user space to kernel space communication

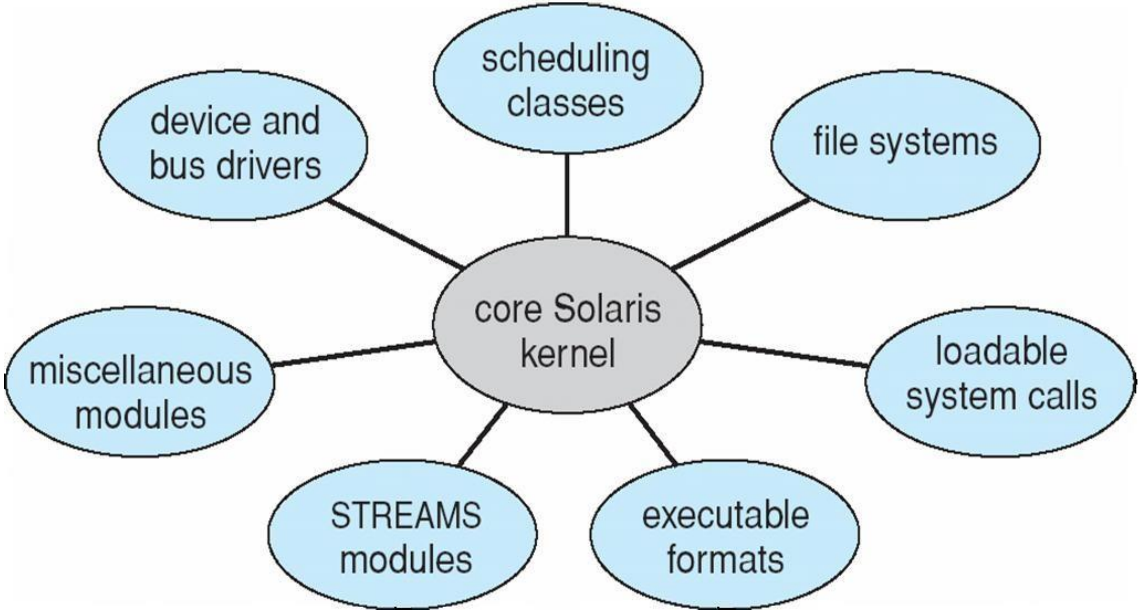
Microkernel System Structure



Modules

- Most modern operating systems implement **loadable kernel modules**
- The kernel has a set of core components and links in additional services via modules, either at boot time or during run time
- The idea is to provide core services for the kernel while other services are implemented dynamically, as the kernel is running.
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- It is similar to layers but with more flexible because any module can call any other module
- Also is similar to the microkernel approach in that the primary module has only core functions and knowledge of how to load and communicate with other modules; but it is more efficient, because modules do not need to invoke message passing.

Solaris Modular Approach

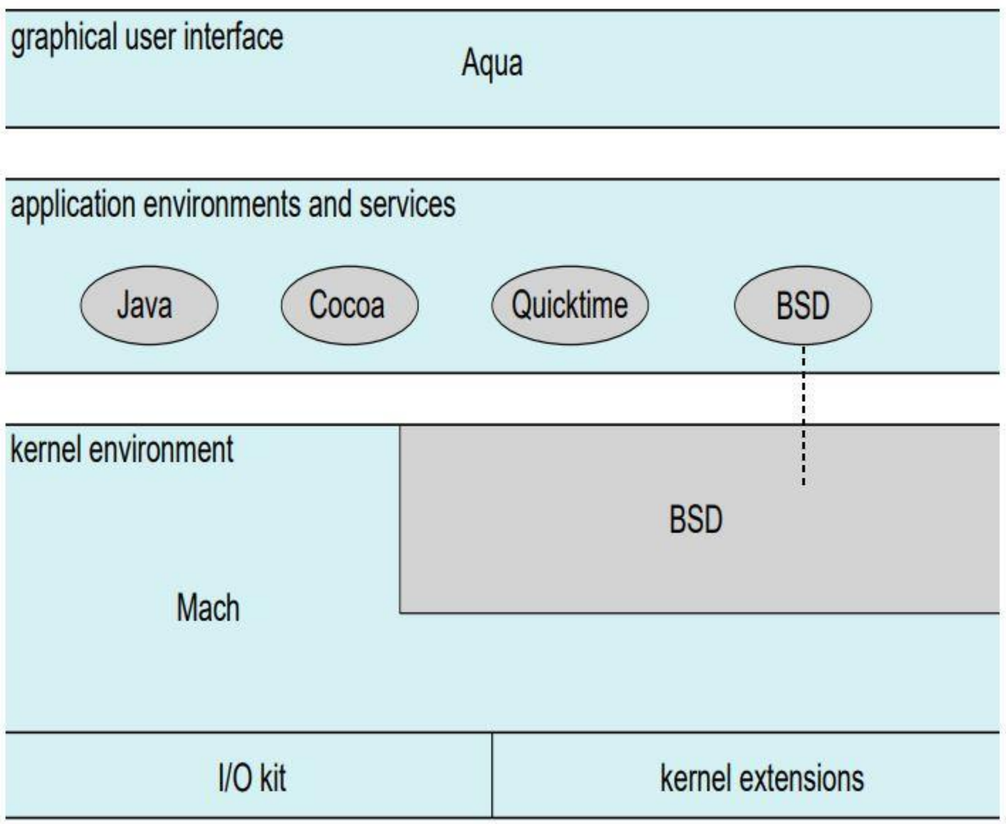


Hybrid Systems

- Most modern operating systems actually not one pure model
  - Hybrid combines multiple approaches to address performance, security, usability needs
  - Linux and Solaris kernels in kernel address space, so monolithic, plus modular for dynamic loading of functionality
  - Windows mostly monolithic, plus microkernel for different subsystem that run as user-mode processes

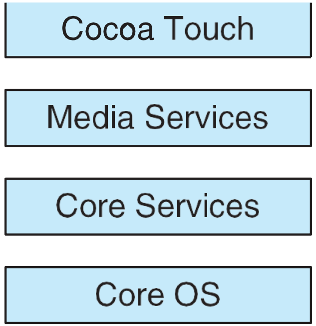
Apple Mac OS X

- It is a layered system, the top **Aqua UI plus Cocoa** programming environment
- Below is kernel consisting of Mach microkernel and BSD Unix parts, plus I/O kit and dynamically loadable modules (called **kernel extensions**)



iOS

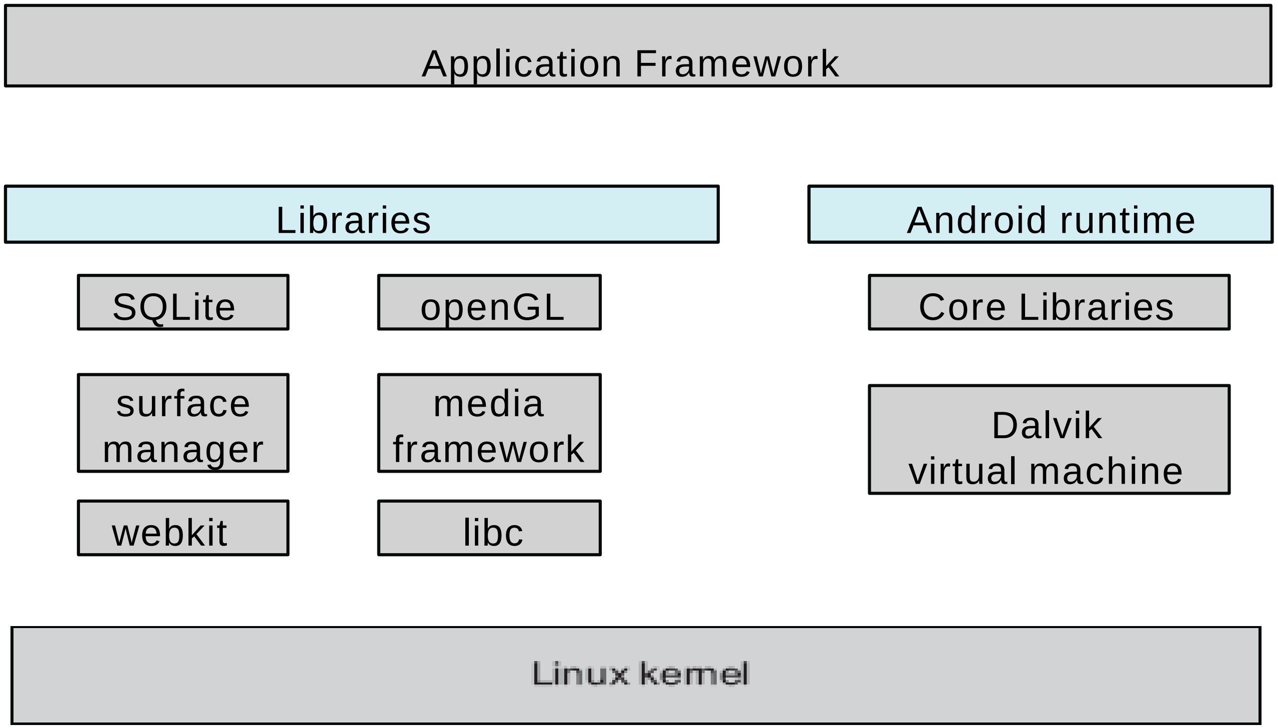
- Apple mobile OS for ***iPhone***, ***iPad***
  - Structured on Mac OS X, added functionality
  - Does not run OS X applications natively
    - Also runs on different CPU architecture
  - **Cocoa Touch** Objective-C API for developing apps
  - **Media services** layer for graphics, audio, video
  - **Core services** provides cloud computing, databases
  - **Core operating system**, based on Mac OS X kernel



Android

- Developed by Open Handset Alliance (mostly Google), it is an Open Source
- Similar stack to IOS; but ios is a closed source.
- Based on Linux kernel but modified
  - Provides process, memory, device-driver management
  - Adds power management
- Runtime environment includes core set of libraries and Dalvik virtual machine
  - Apps developed in Java plus Android API
    - Java class files compiled to Java bytecode then translated to executable then runs in Dalvik VM
- Libraries include frameworks for web browser (webkit), database (SQLite), multimedia, smaller libc

Android Architecture



End of Chapter 2