

# **Operating System 1**

**Dr. Faris Llwaah**

**University of Mosul  
College of Computer Sciences and Mathematics  
Department of Computing Sciences**



# Operating System Structure

In this lecture we address the following topics:

1. Simple structure
2. Layered approach.
3. Microkernels.
4. Modules.
5. Virtual Machines.

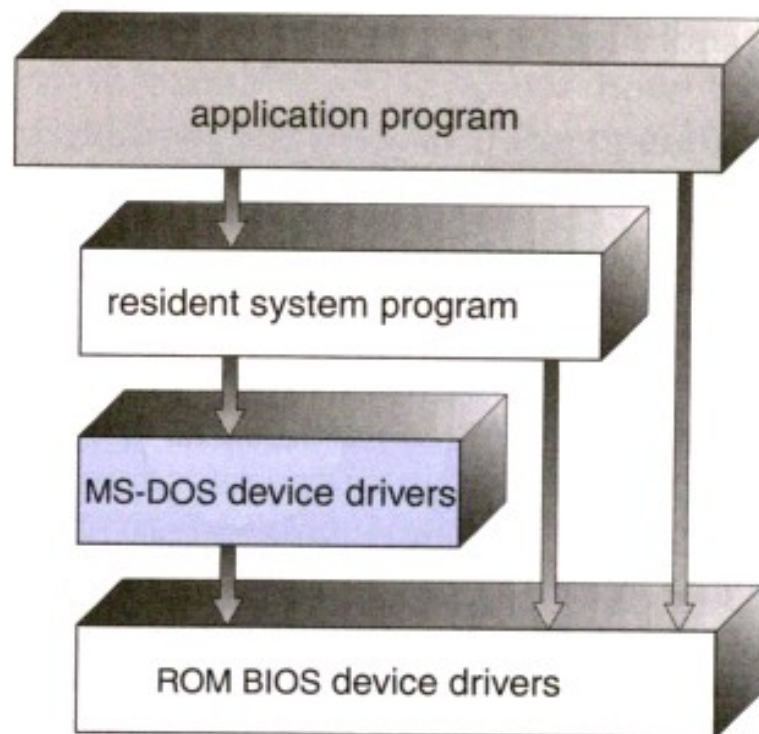
# ❑ Operating System Structure

- General-purpose OS is very large program
- Various ways to structure ones
  - Simple structure – MS-DOS
  - More complex -- UNIX
  - Layered – an abstraction
  - Microkernel -Mach

# ❑ Simple Structure

❑ For example of simple structure, it is MS-DOS layer structure

- MS-DOS was created to provide the most functionality in the least space. Not divided into modules MS-DOS has some structure. But its interfaces and levels of functionality are not well separated. No dual mode existed for Intel 8088. MS-DOS was developed for 8086. Direct access to hardware is allowed.

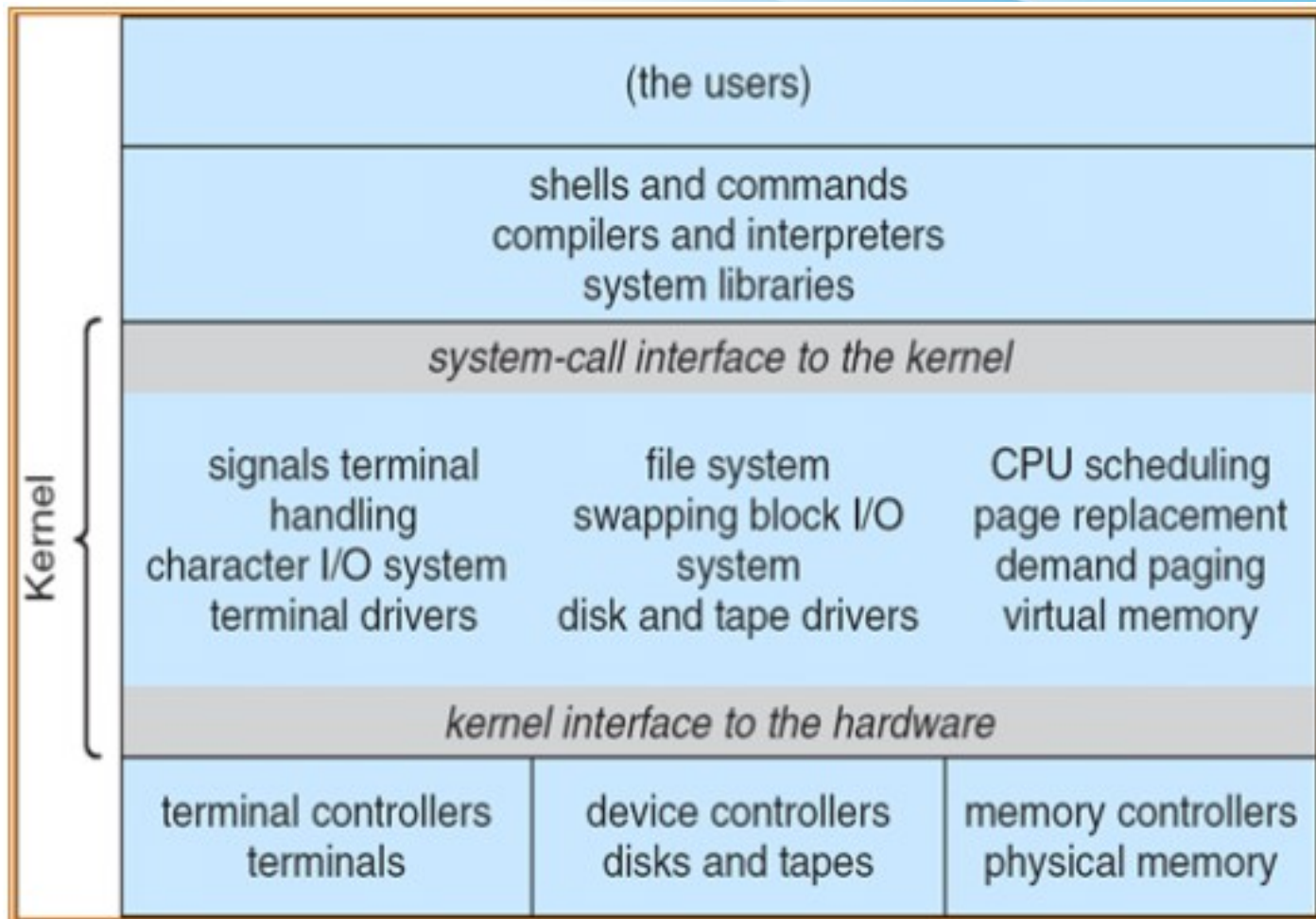


# ❑ UNIX partially-layered structure

Unix- the original UNIX operating system had limited structuring. The UNIX OS consist of two separable parts:

- Systems programs
- The Kernel
  - \* Consist 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, large number of functions for one level.

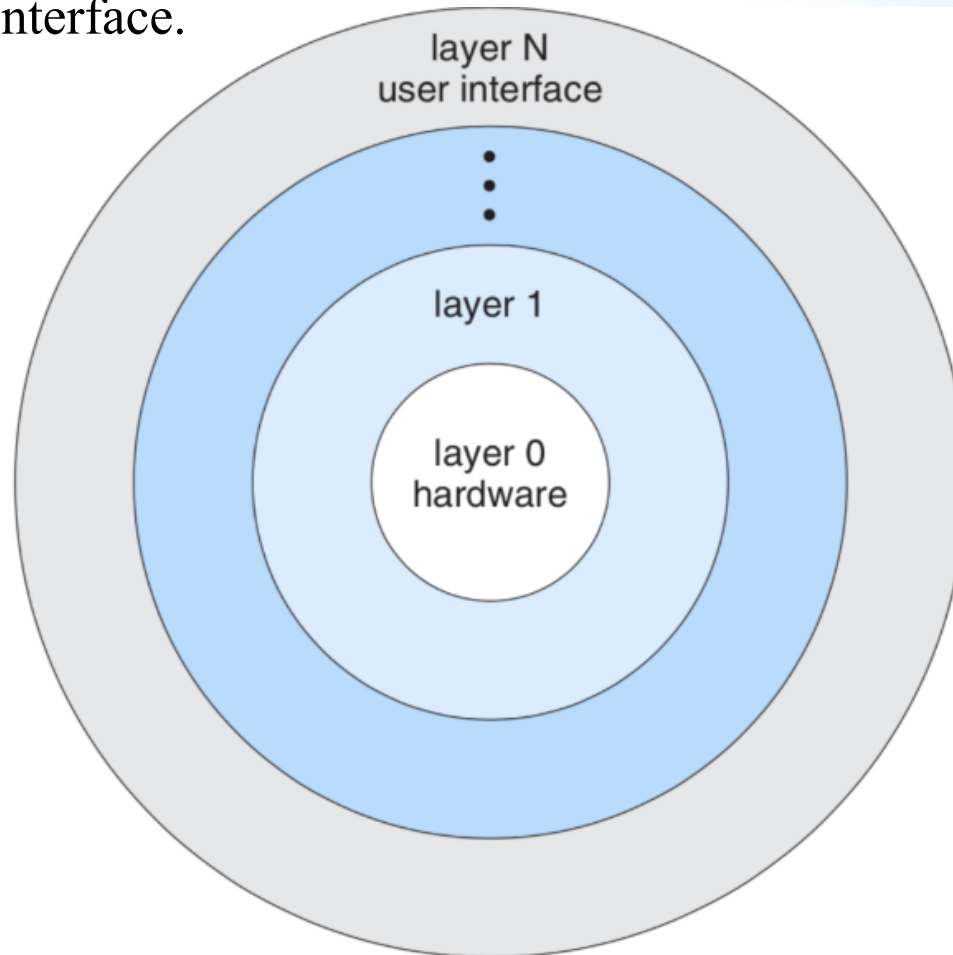
# □ UNIX system structure





## □ Layered approach

The operating system can be described in many ways. One method is the **layered approach**, in which the operating system is broken into a number of layers as shown in the previous figure. The bottom layer (layer 0) is the hardware. The highest layer (layer  $N$ ) is the user interface.





- ❑ The main advantages of the layered approach is simplicity of construction and debugging.
- The layers are selected so that each uses functions and services of only lower-level layers.
- If an error is found during the debugging of a particular layer, the error must be on that layer, because the layers below it are already debugged.
- For instance, when a user program executes an I/O operation, it executes a system call that is trapped to the I/O layer, which calls the memory-management layer, which in turn calls the CPU-scheduling layer, which is then passed to the hardware.

❑ The disadvantages of the layered approach are

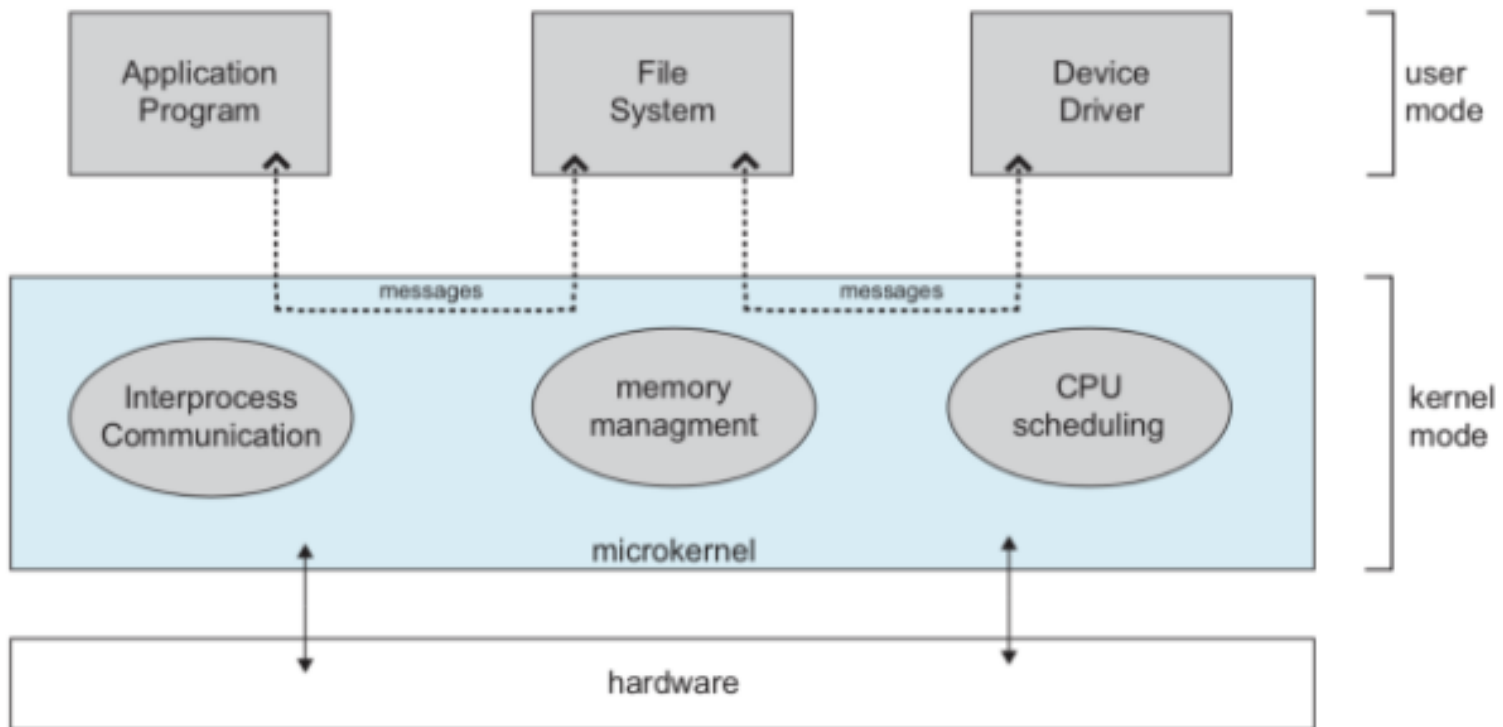
- Precise definition of layers. For example, memory manager requires device driver of backing store (due to virtual memory), as well the device driver requires CPU scheduler (since if the driver waits for IO, another task should be scheduled). And CPU scheduler may require virtual memory for large amount of information of some processes.
- Less efficiency, due to the number of layers a request should pass
- Each layer adds overhead to the system call. The net result is a system call that takes longer than it does one on a no layered system.

# ❑ Microkernel System Structure

- Moves as much from the kernel into user space
- **Mach** 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

# ❑ Microkernels

This method structures the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result is a smaller kernel.



## ❑ What the Kernel does?

- Microkernels provide minimal process and memory management.
- The main function of the microkernel is to provide communication between the client program and the various services that are also running in user space. So, the resulting operating system is easier to port from one hardware design to another.
- The microkernel also provides more security and reliability, since most services are running as user rather than kernel processes.

## ❑ The advantages of Kernel

- Extensibility of the OS
- Portability
- Potential for making distributed services
- More security and reliability, i.e. service failure doesn't destroy OS.

## ❑ The disadvantage of Kernel

- Performance loss due to Inter-process communication (IPC)
- For example of kernel's OS Mach, QNX, L4, the first release of windows NT.....

## ❏ Modules

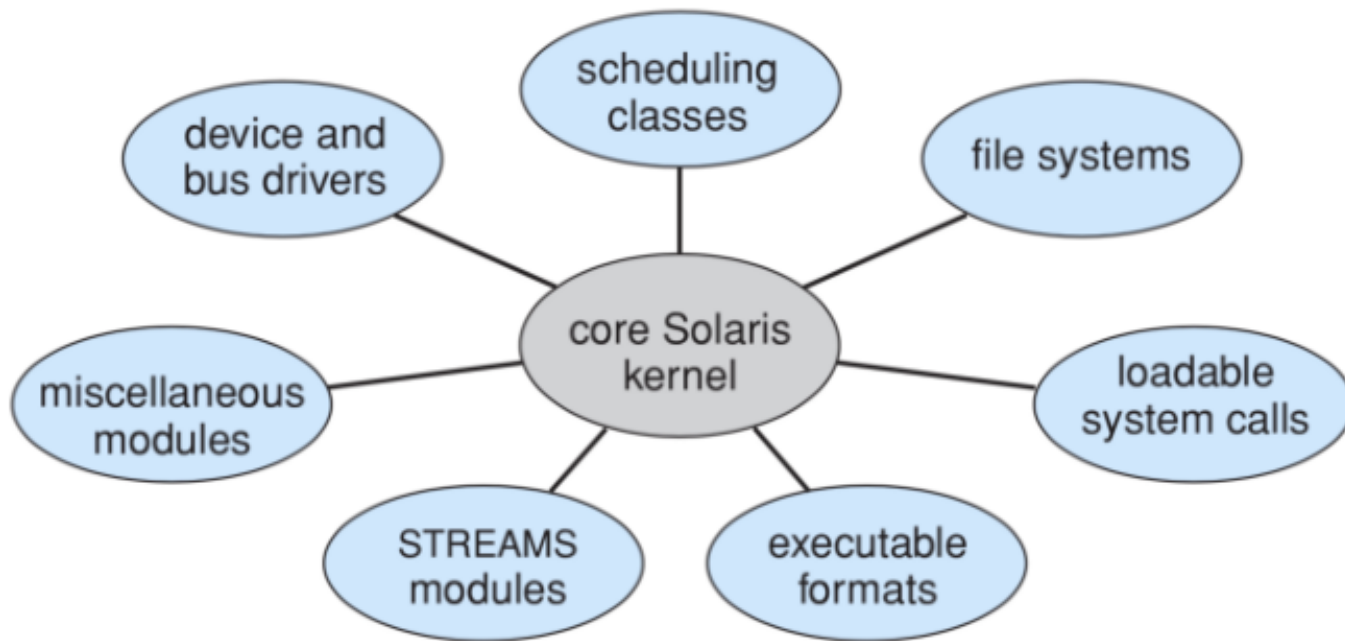
One of the best current methodologies is to design operating-system which involves using **loadable kernel modules**. Here, the kernel has a set of core components and links in additional services via modules, either at boot time or during run time.


- The idea of this design is for the kernel to provide core services while other services are implemented dynamically, as the kernel is running.
- Linking services dynamically is preferable to adding new features directly to the kernel, which would require recompiling the kernel every time a change was made.
- The overall result resembles a layered system in that each kernel section has defined, protected interfaces; but it is more flexible than a layered system, because any module can call any other module. The approach is also 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 in order to communicate.



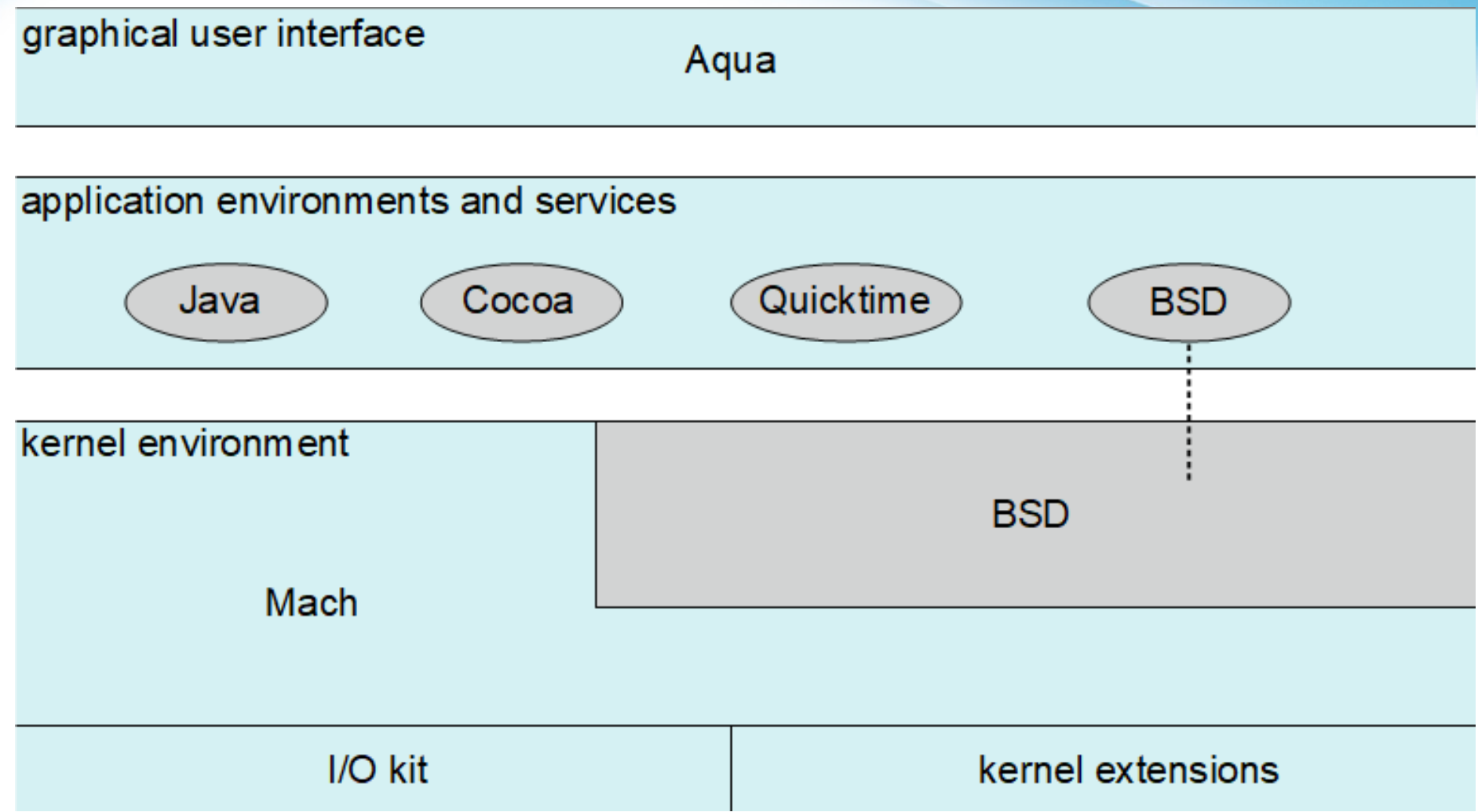
# ☐ Solaris Modular Approach

- The Solaris operating system structure, shown in diagram, is organised around a core kernel



- 
- Scheduling classes
  - File systems
  - Loadable system calls
  - Executable formats
  - STREAMS modules
  - Miscellaneous
  - Device and bus drivers

# ❑ Mac OS X Structure



# □ 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 (ARM vs. Intel)
- **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

Cocoa Touch

Media Services

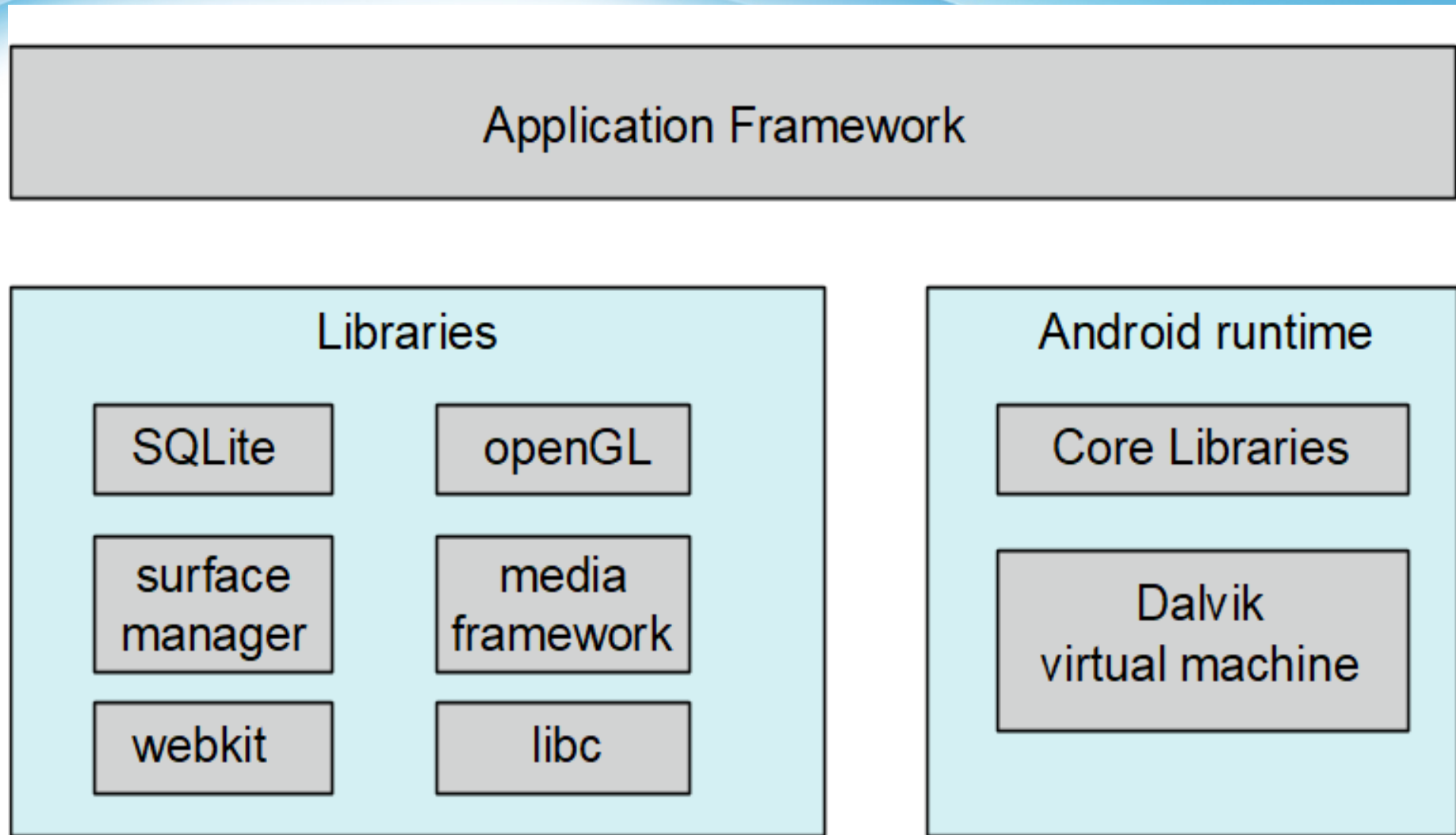
Core Services

Core OS

## ❏ Android

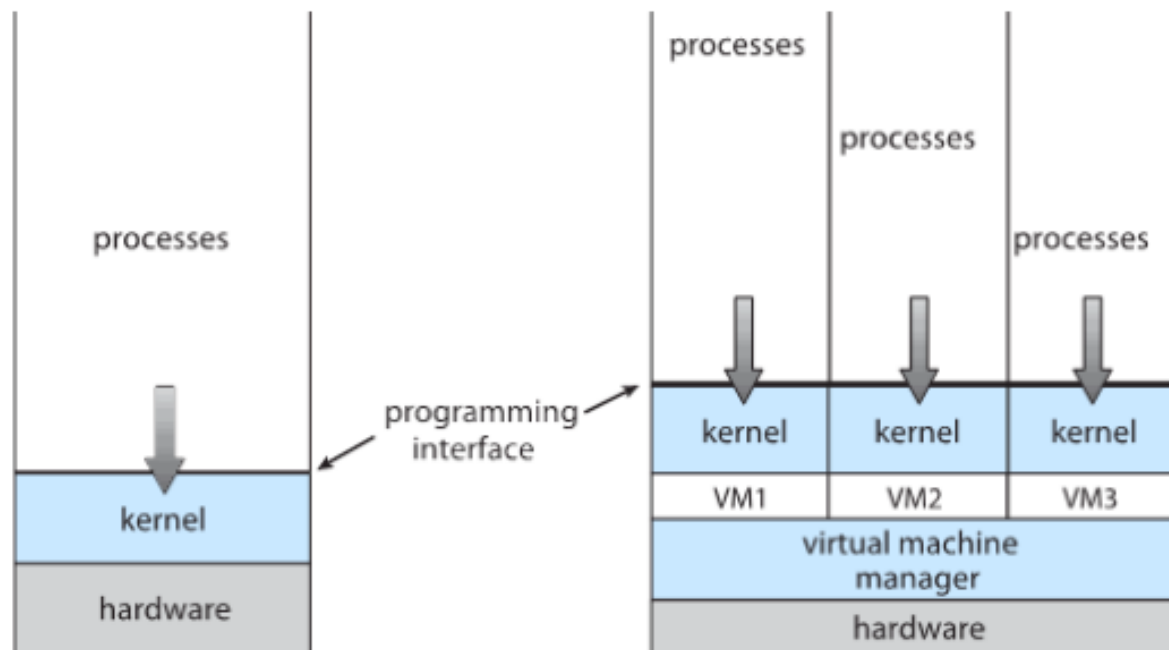
- Developed by Open Handset Alliance (mostly Google)
  - Open Source
- Similar stack to IOS
- Based on Linux kernel but modified
  - Provides process, memory, device-driver management
  - Adds power management
- Runtime environment includes core set of libraries and Dalvik virtual machine
  - Apps developed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

# ❑ Android Architecture



## ❑ Virtual Machines

A virtual machine is a computer file, typically called an image, that behaves like an actual computer. In other words, creating a computer within a computer. It runs in a window, much like any other program, giving the end user the same experience on a virtual machine as they would have on the host operating system itself. In computing, a **virtual machine (VM)** is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software, or a combination.





## ❑ **Benefits**

- VMs are completely isolated, no protection problem
- No direct sharing of resources
- Sharing the same HW among different environments with different operation systems running concurrently
- It is perfect vehicle for OS and development

## ❑ **Main difficulty**

- Disk space

## ❑ **Implementation**

- User mode: Virtual user mode, Virtual Kernel mode
- Kernel mode

❑ **The underlying OS may be structured as layered, Kernel,.....**

❑ **Major difference is time**

- Real I/O might take 100ms
- Virtual I/O might take less time or more time