Structures of Operating Systems

Operating Systems

School of Data & Computer Science
Sun Yat-sen University

Lecture Notes: os_sysu@163.com
Instructor: Guoyang Cai
email: isscgy@mail.sysu.edu.cn

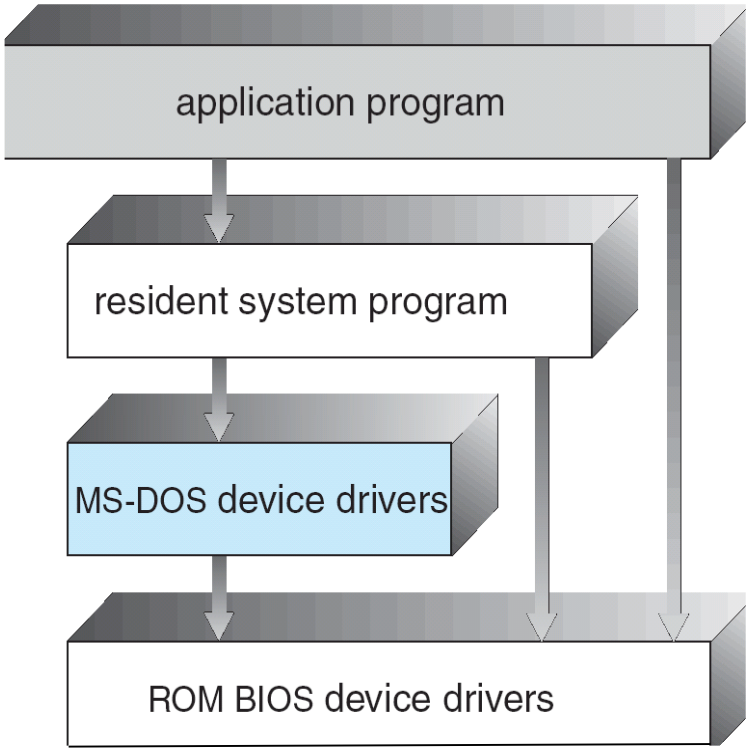
Contents

Monolithic Operating System

- Basic Structure of a monolithic operating system (宏内核)
  - *Application programs* that invokes the requested system services.
  - A set of *system services* that carry out the operating system procedures/calls.
  - A set of *utility procedures* that help the system services.
  - All of the functionality of the kernel are placed into a single, static binary file that runs in a single address space.
- MS-DOS
  - MS-DOS is written to provide the most functionality in the least space.
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated.
    - not divided into modules.

Monolithic Operating System

- MS-DOS
  - MS-DOS layer structure.

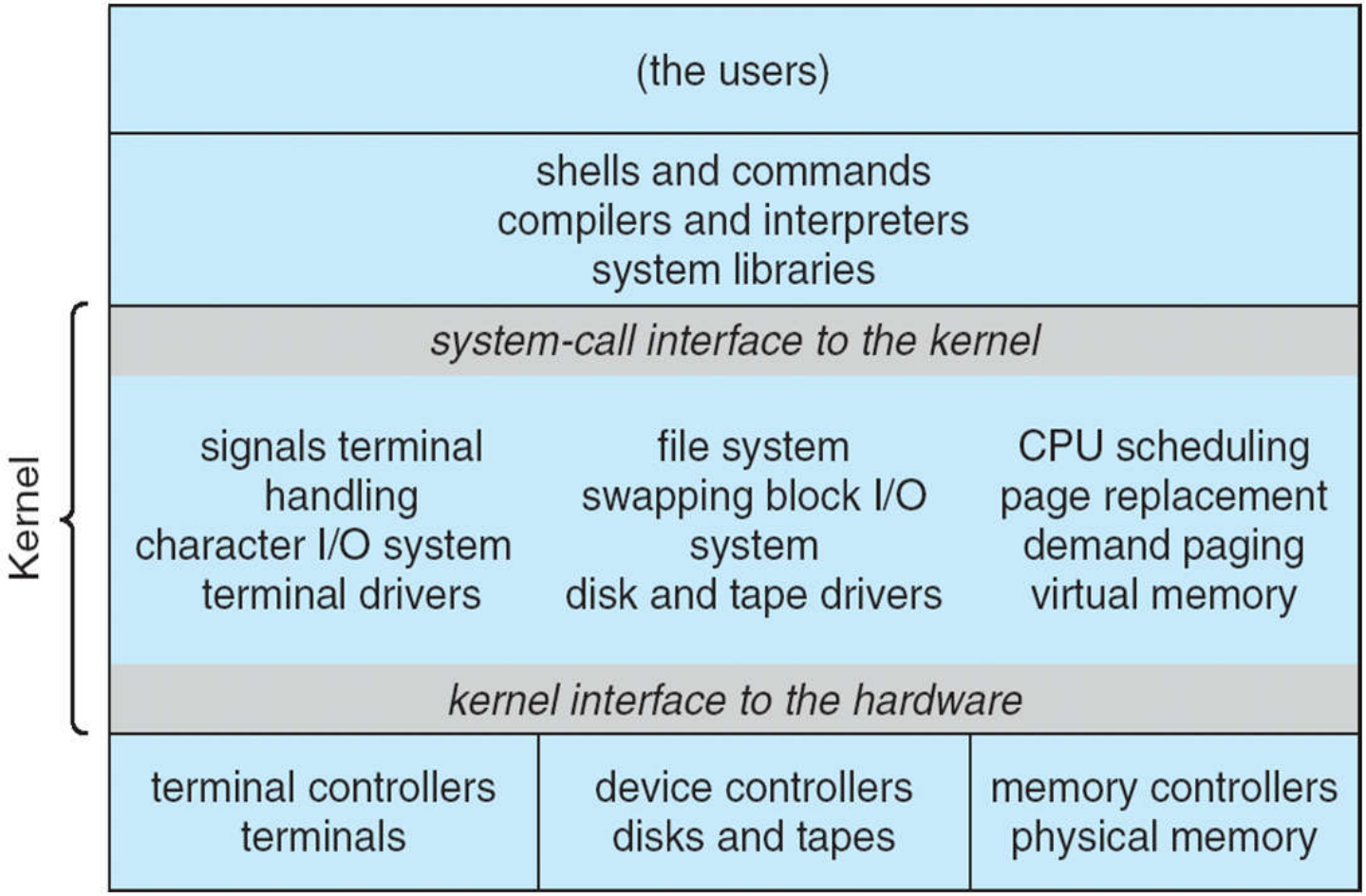


Monolithic Operating System

- UNIX
  - UNIX is limited by hardware functionality, the original UNIX OS had limited structuring.
  - UNIX consists of two separable parts:
    - *Systems Programs*
    - *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.

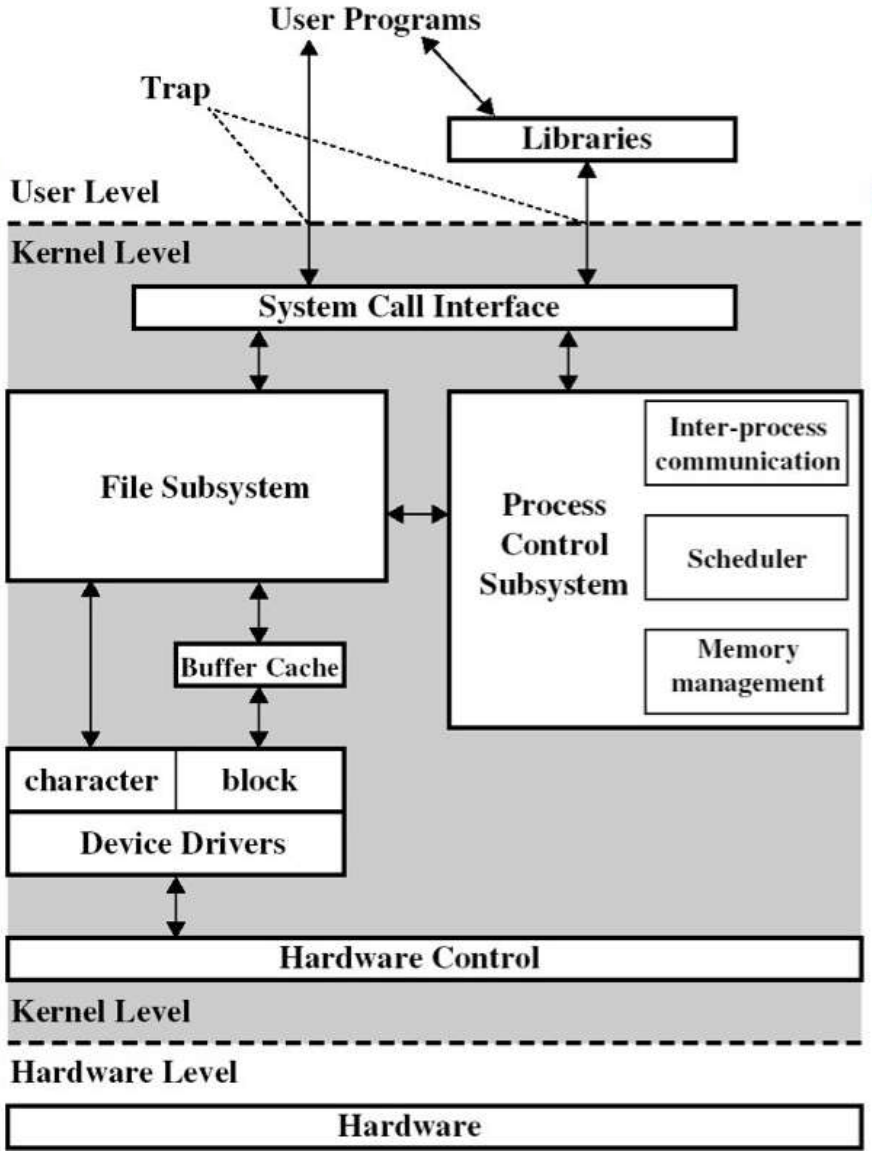
Monolithic Operating System

- UNIX
  - Traditional UNIX system structure.



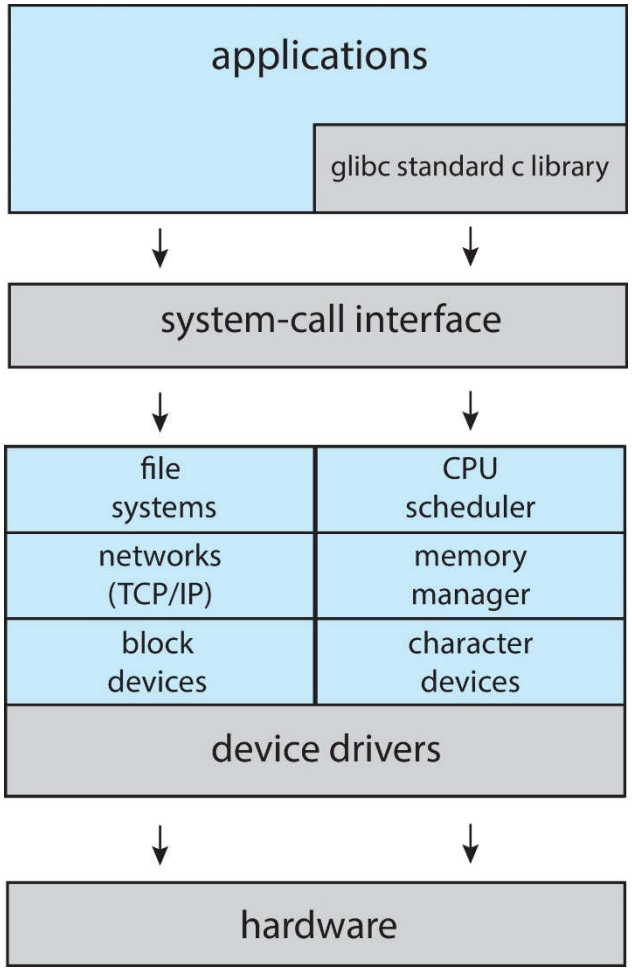
Monolithic Operating System

- UNIX
  - Traditional UNIX kernel



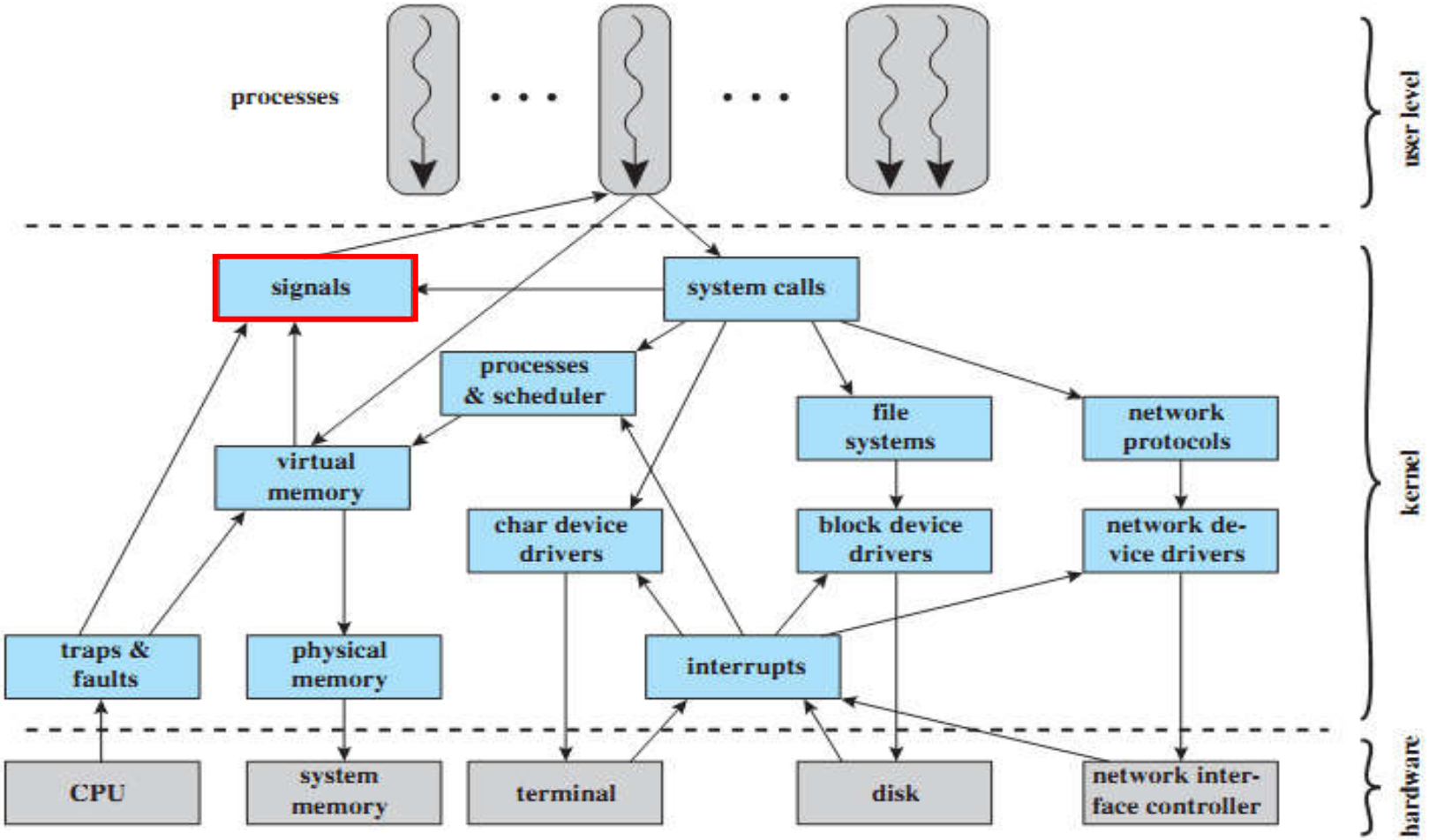
Monolithic Operating System

- Linux
  - Linux system structure.



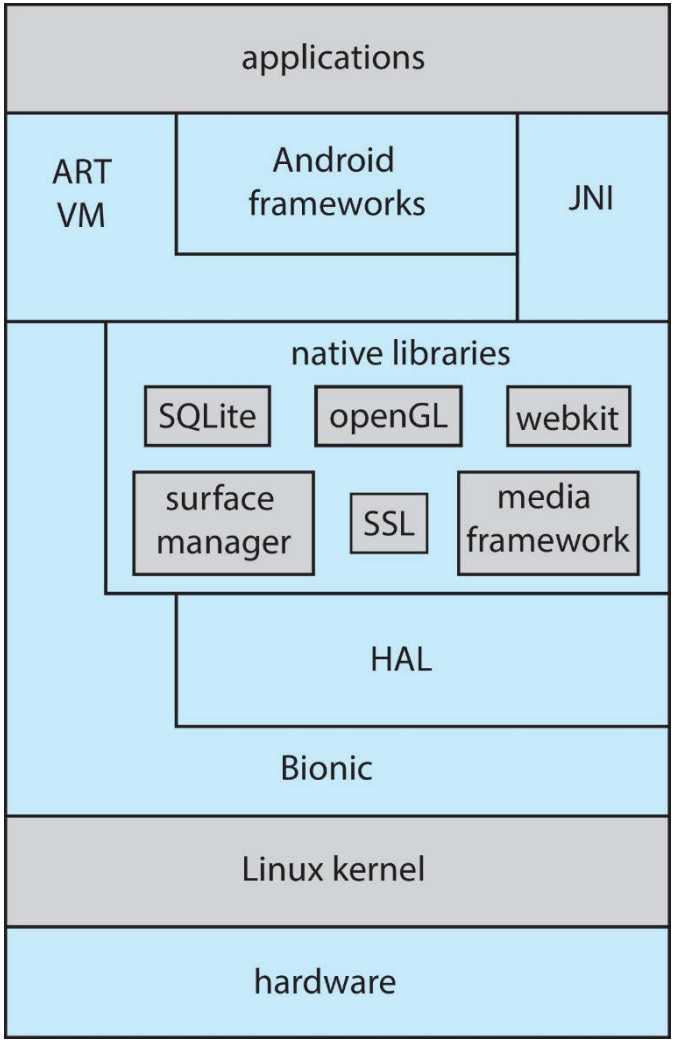
Monolithic Operating System

- Linux
  - Linux kernel components.



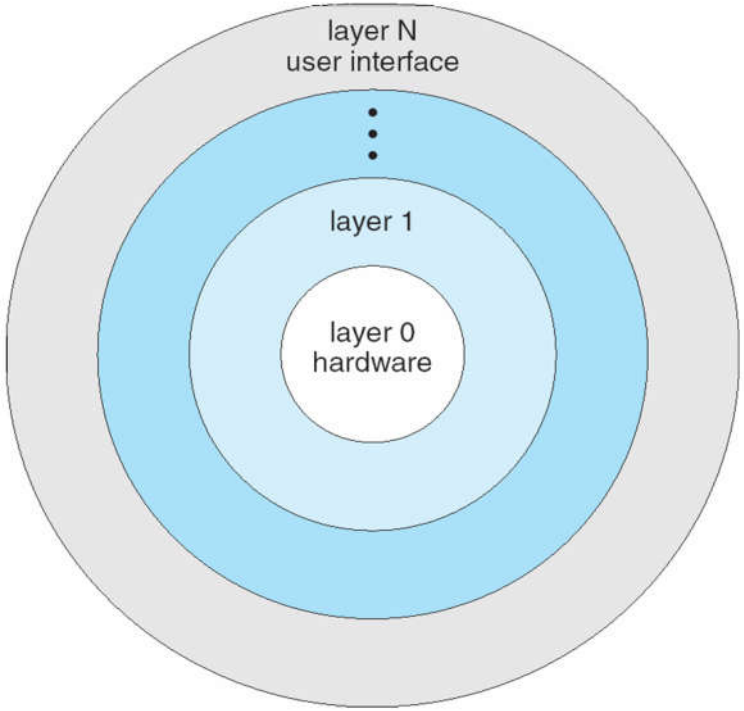
Monolithic Operating System

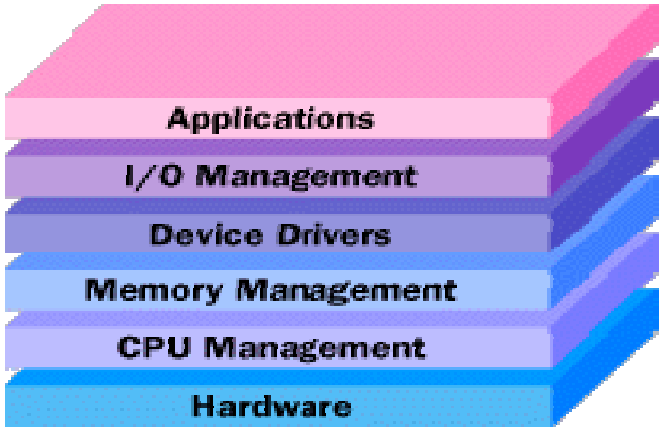
- Google's Android.



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.



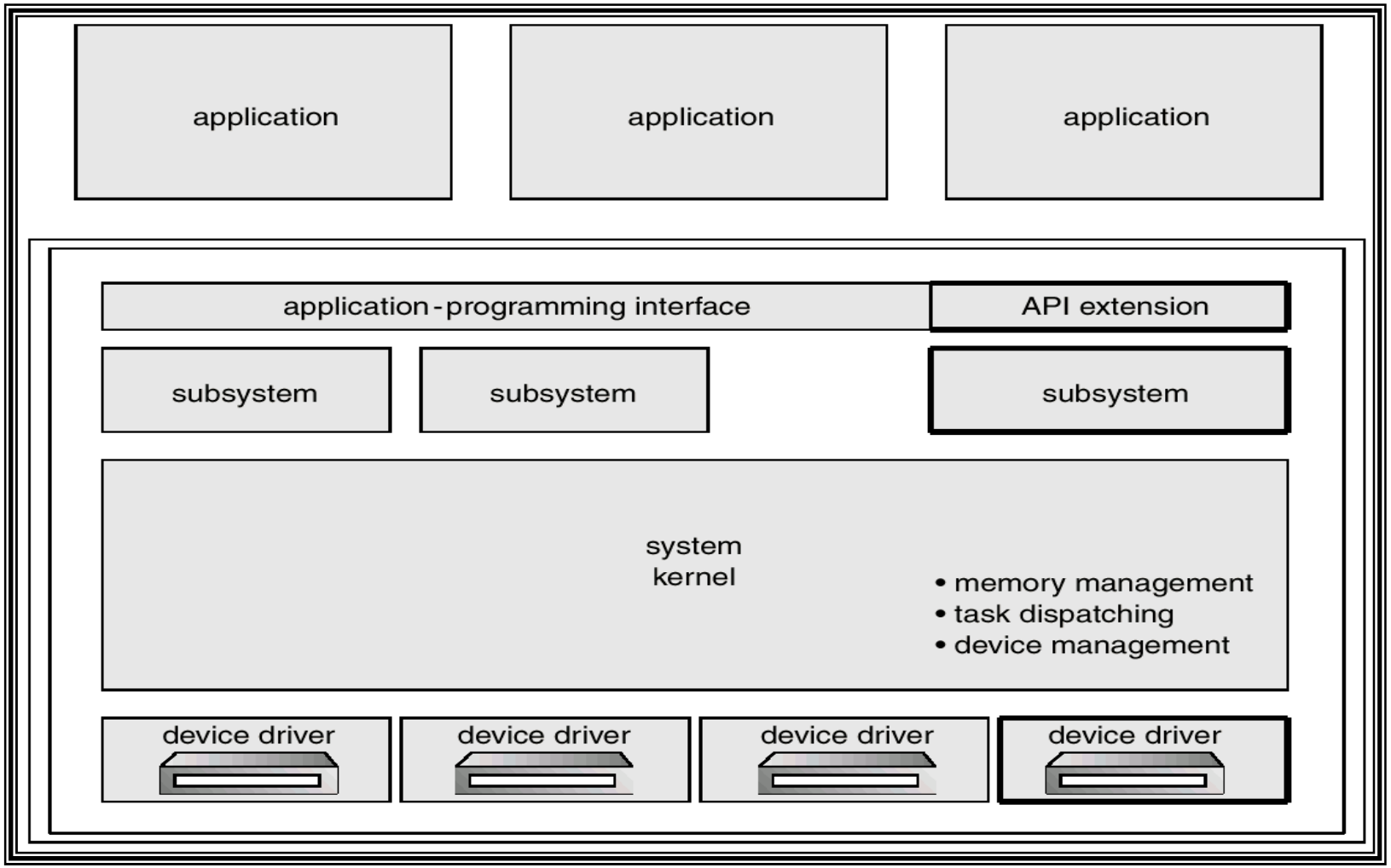


Layered Approach

- Older Windows System Layers.

Layered Approach

- OS/2 Layer Structure.

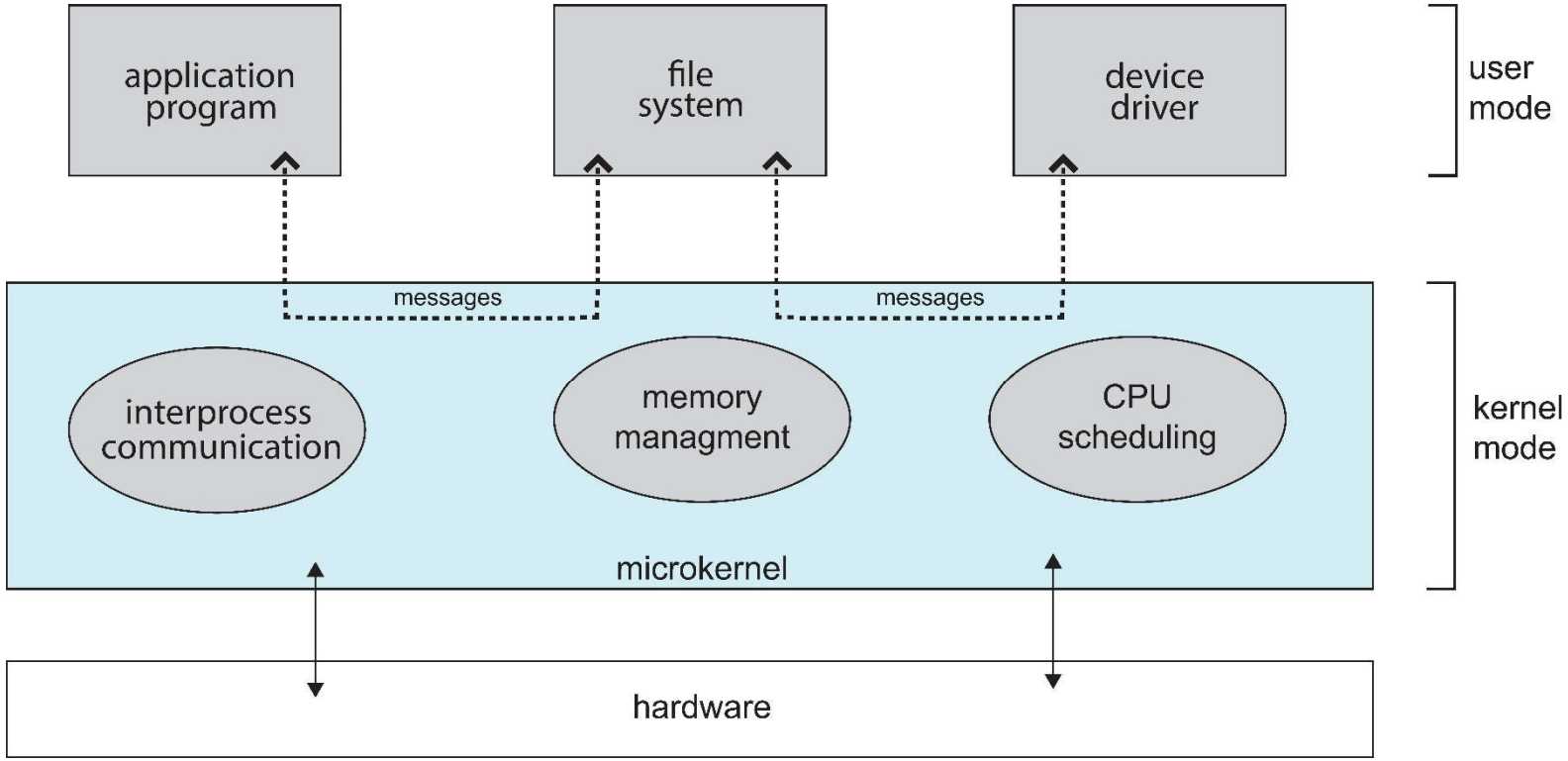


Microkernel System Structure

- Microkernel systems (微内核) move as much functionality as possible from the kernel into “user” space.
  - As UNIX expanded, the kernel became large and difficult to manage.
  - The *Mach 3.0* operating system (CMU, 1985-1994) modularized the kernel using the *microkernel* approach.
  - The result is a smaller kernel. Only a few essential functions stay in the kernel:
    - primitive memory management (address space)
    - I/O and interrupt management
    - InterProcess Communication (IPC)
    - basic scheduling.
  - Other OS services are provided by processes running in user mode (vertical servers):
    - device drivers, file system, virtual memory, …
  - Communication takes place between user modules using message passing.

Microkernel System Structure

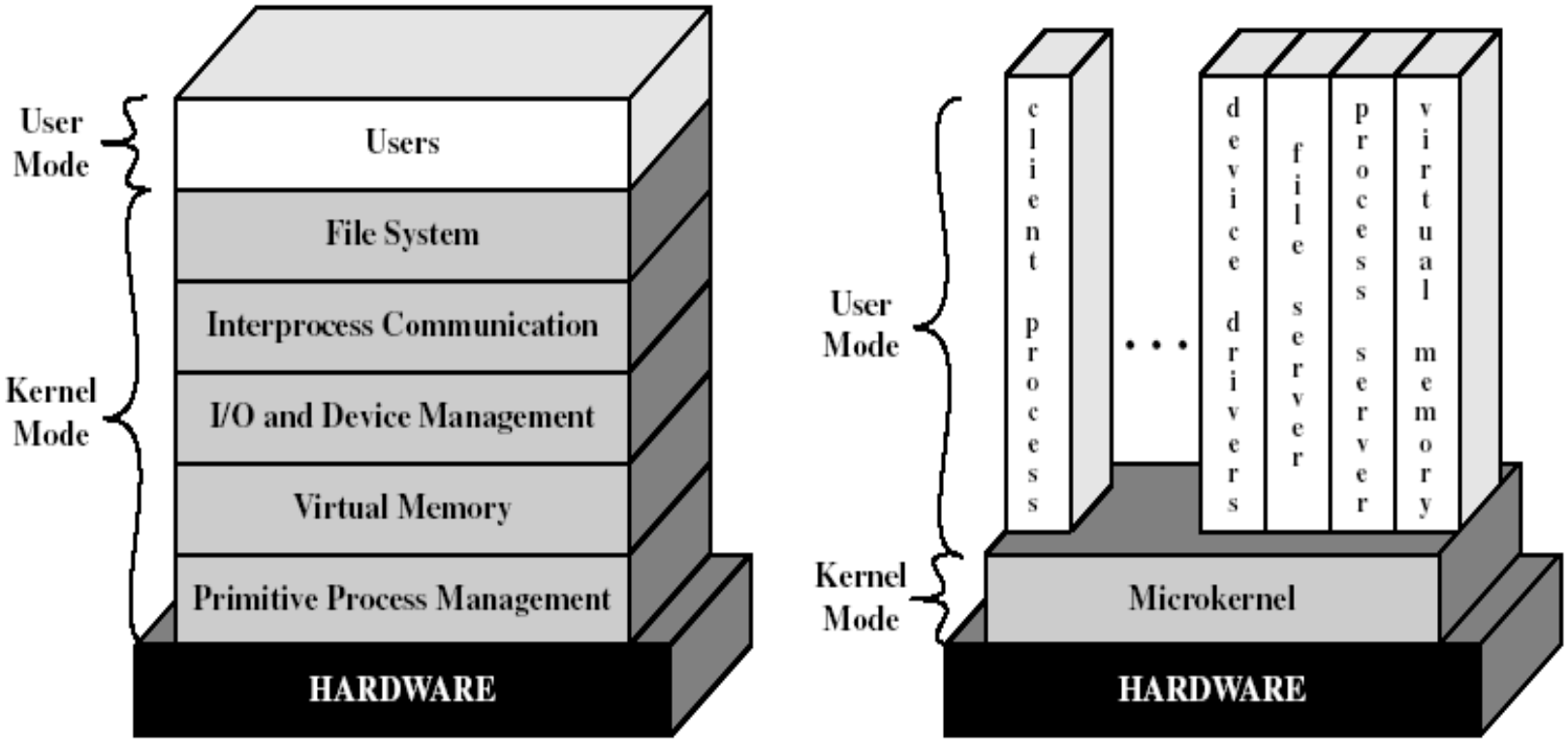
- Performance overheads are caused by replacing service calls with message exchanges between processes.



Architecture of a typical microkernel

Microkernel System Structure

- Layered vs. Microkernel Architecture.

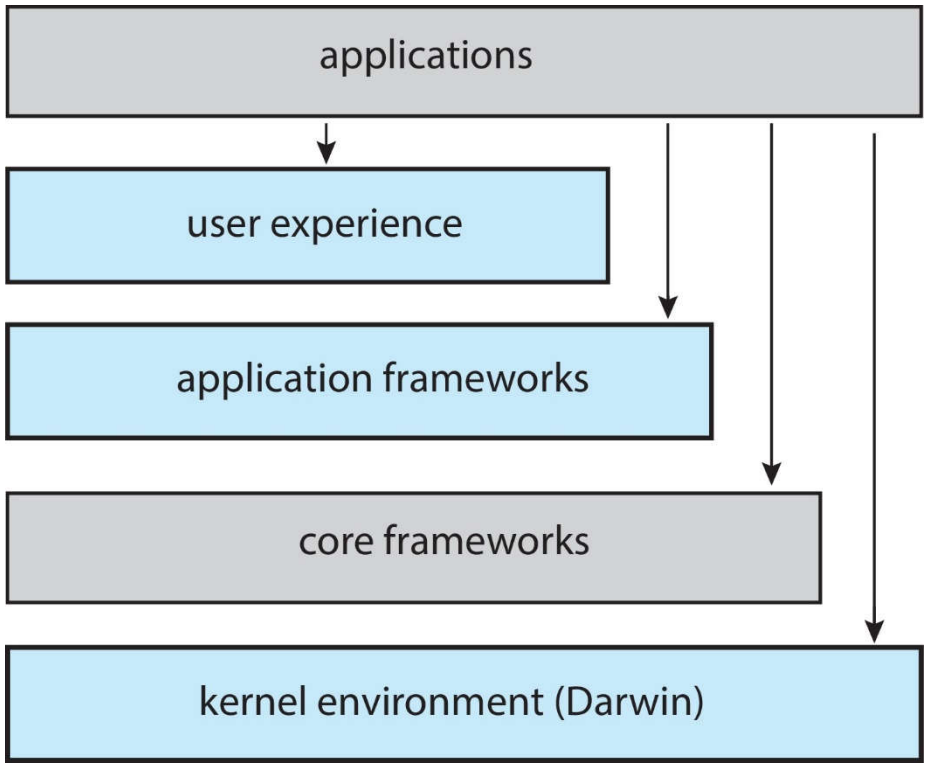


Microkernel System Structure

- Benefits of a Microkernel Organization
  - Extensibility/Reliability
    - easier to extend a microkernel
    - easier to port the operating system to new hardware architectures
    - more reliable (less code is running in kernel mode)
    - more secure
    - small microkernel can be rigorously tested.
  - Portability
    - Changes needed to port the system to a new processor is done in the microkernel, not in the other services.
  - Distributed system support
    - Message are sent without knowing what the target machine is.
  - Object-oriented operating system
    - Components are objects with clearly defined interfaces that can be interconnected to form software.

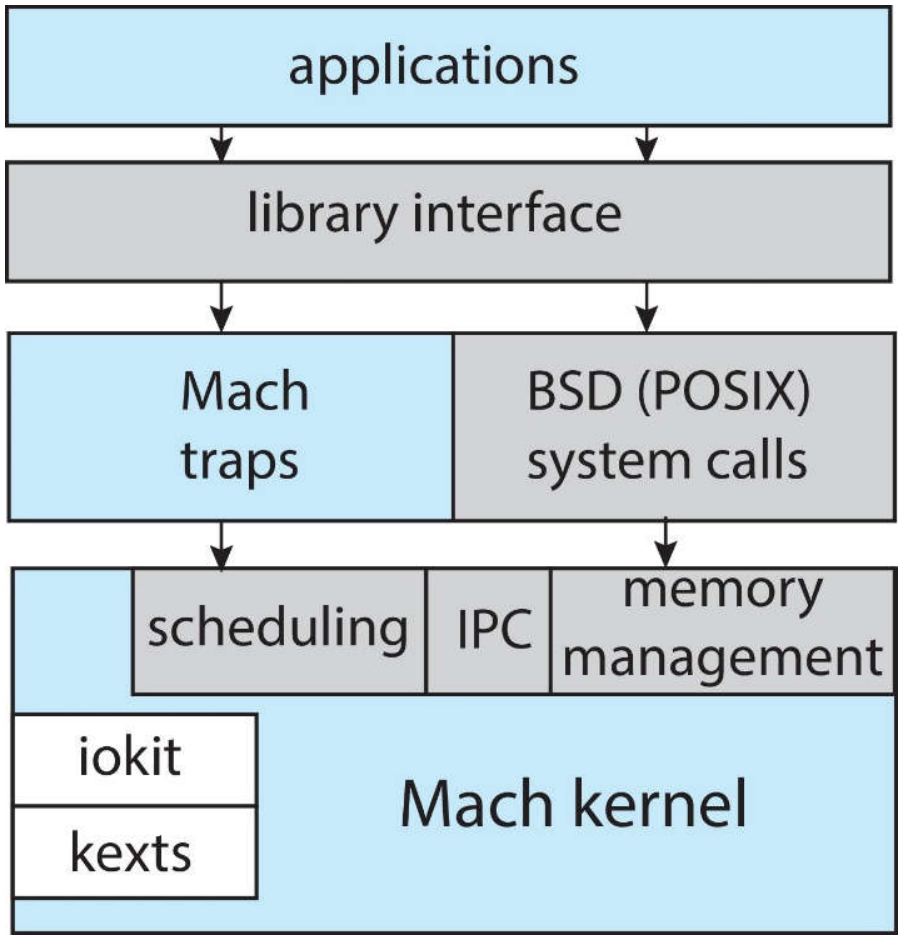
Microkernel System Structure

- MacOS System Structure
  - Architecture of Apple's macOS and iOS operating systems.



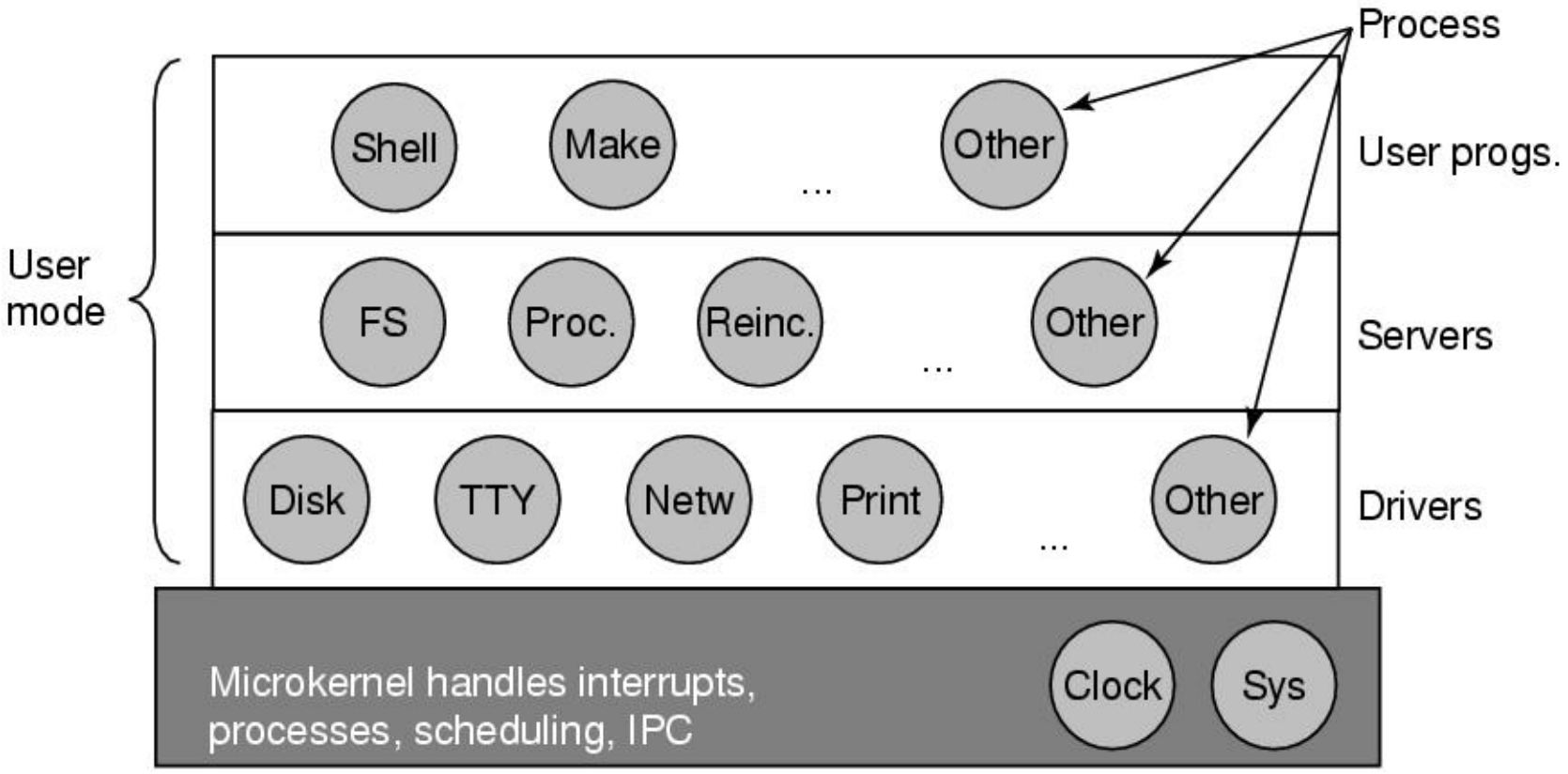
Microkernel System Structure

- MacOS System Structure
  - The structure of Darwin.



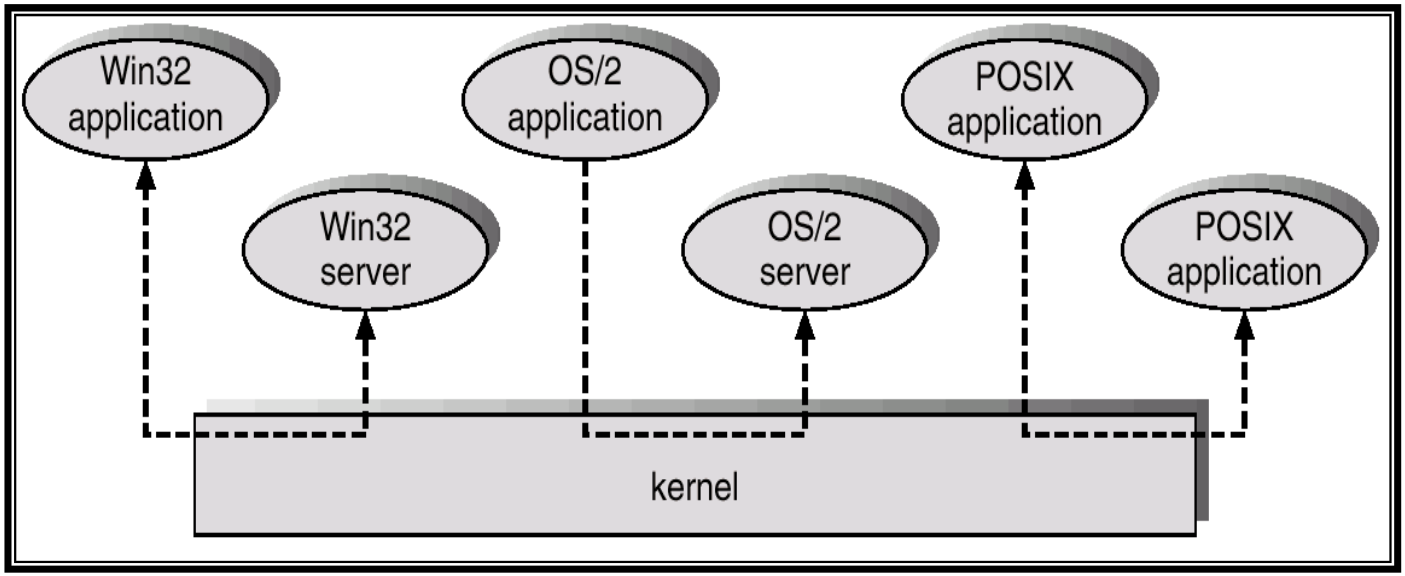
Microkernel System Structure

- MINIX 3 Structure
  - *Linus Torvalds* vs. *Andrew Tanenbaum*: The controversy over monolithic and microkernel



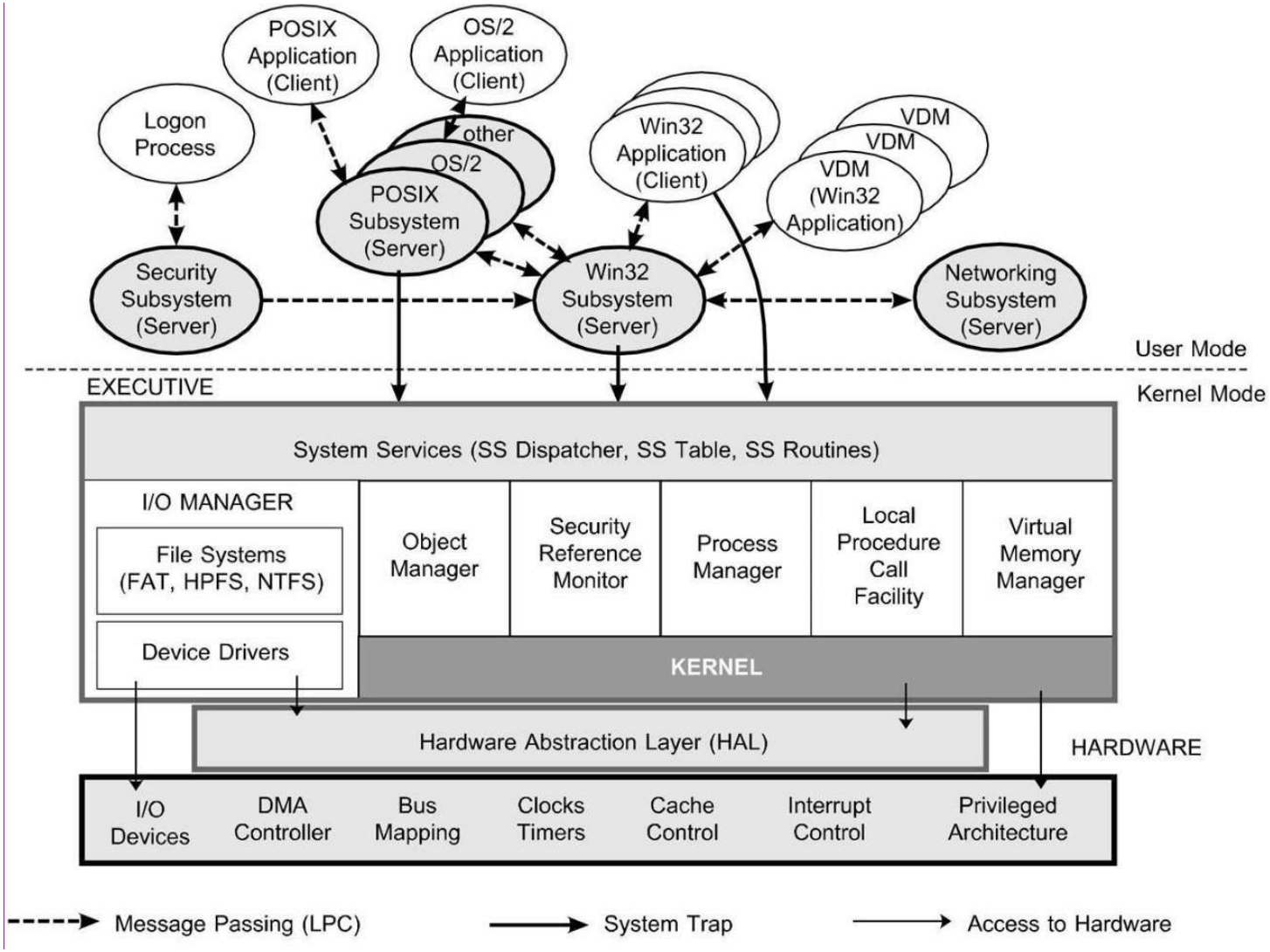
Microkernel System Structure

- Windows NT Client/Server Structure.



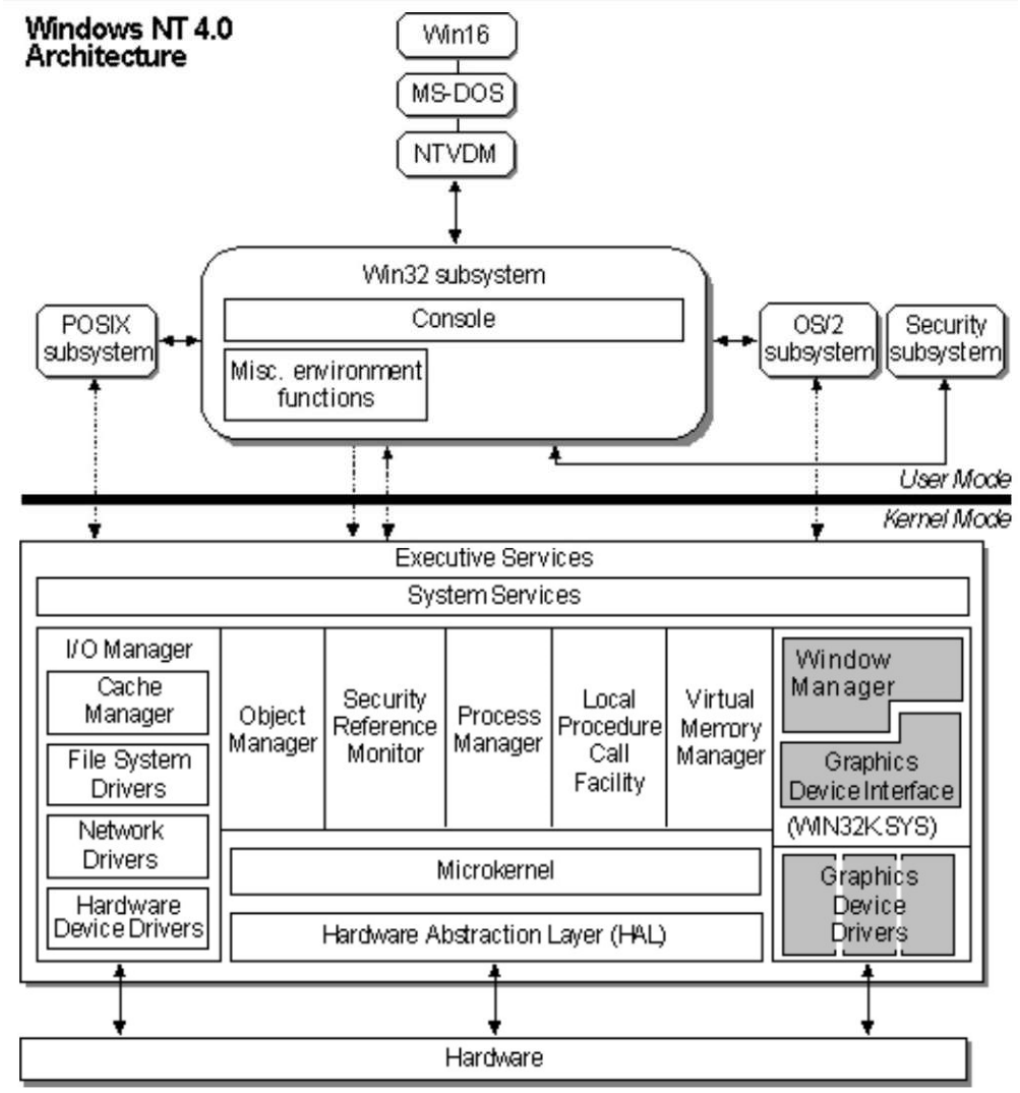
Microkernel System Structure

- Windows NT 4.0 Architecture.



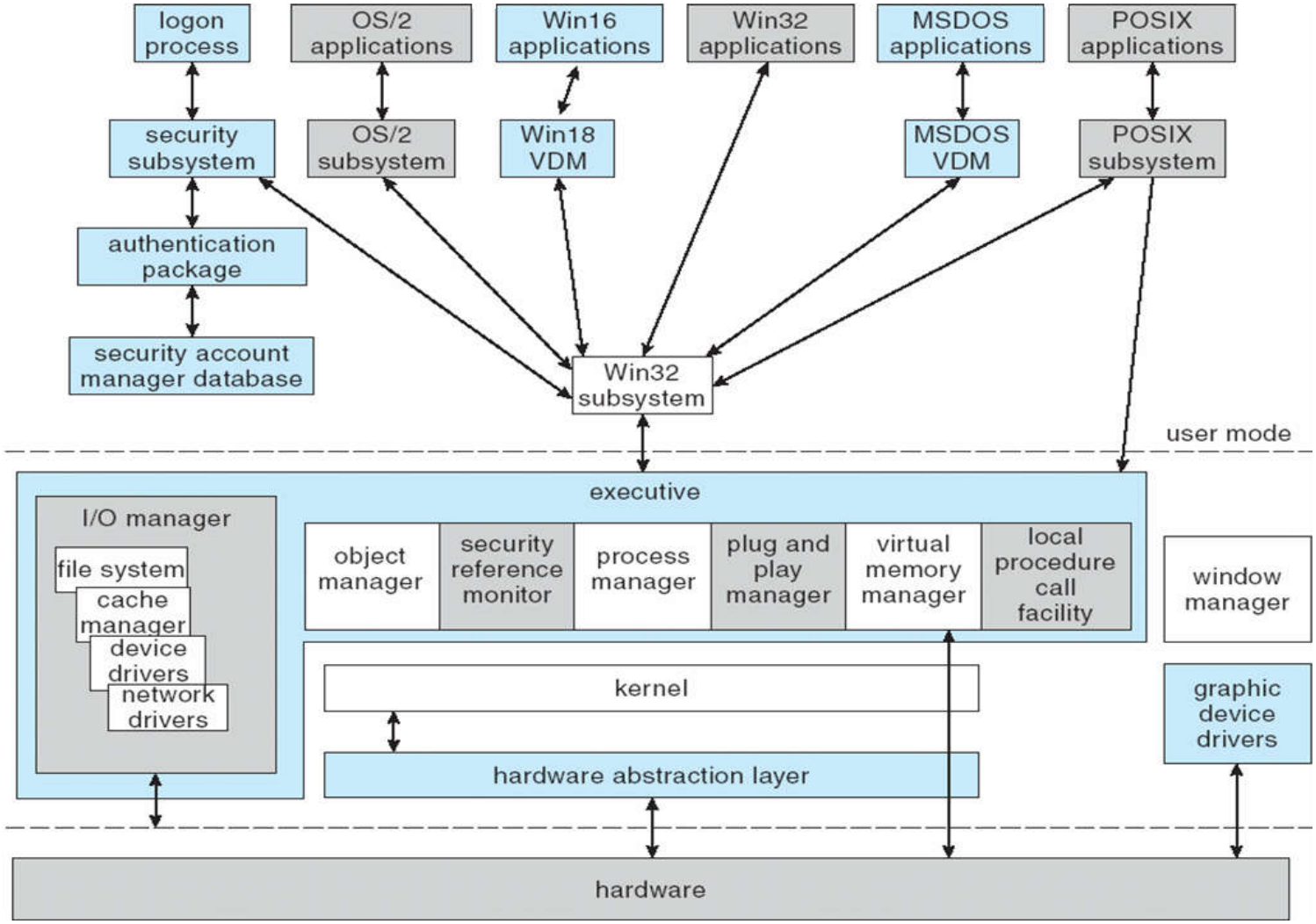
Microkernel System Structure

- Windows NT 4.0 Architecture.



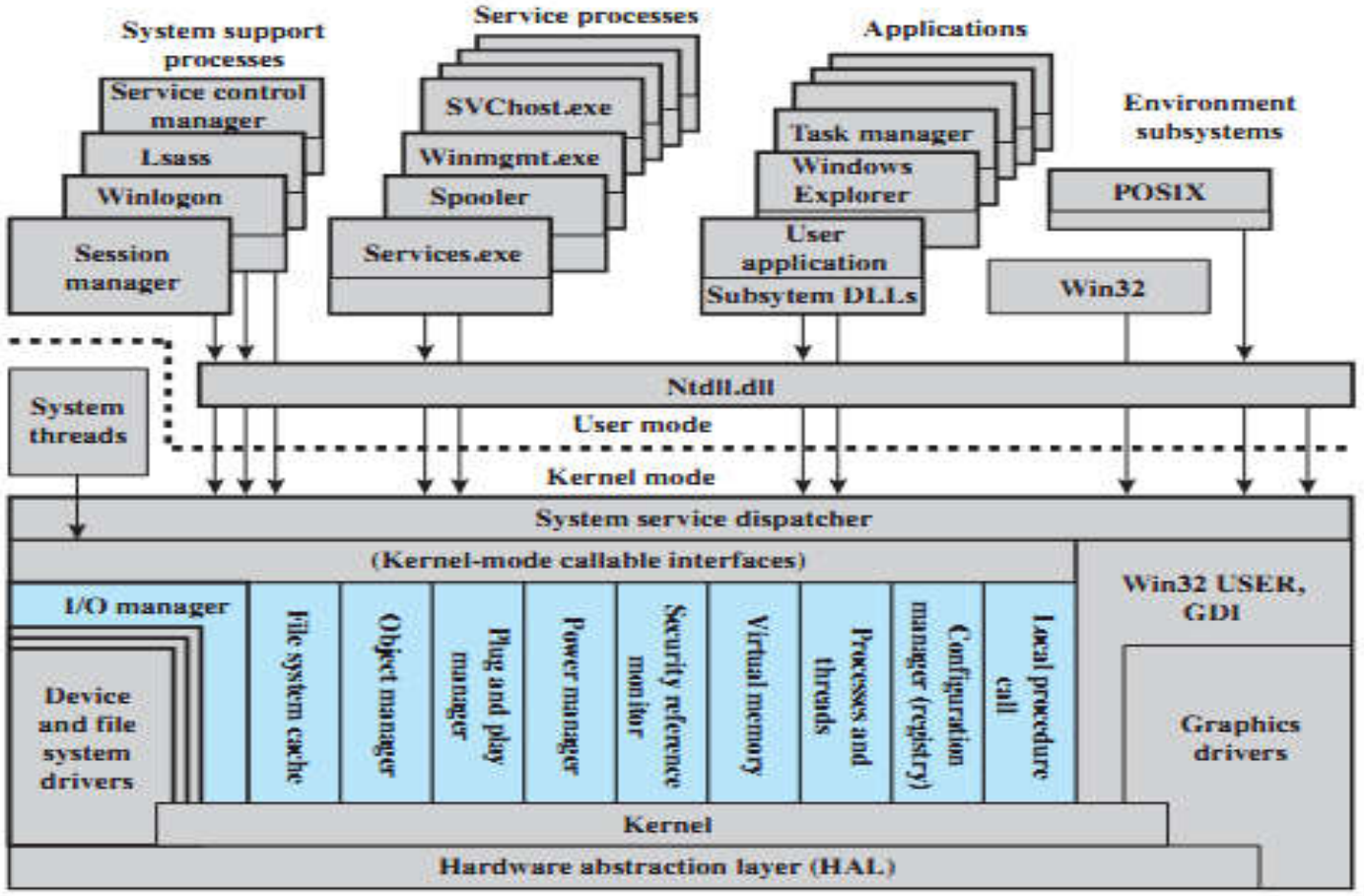
Microkernel System Structure

- Windows XP Architecture.



Microkernel System Structure

- Windows 7.0 Architecture.

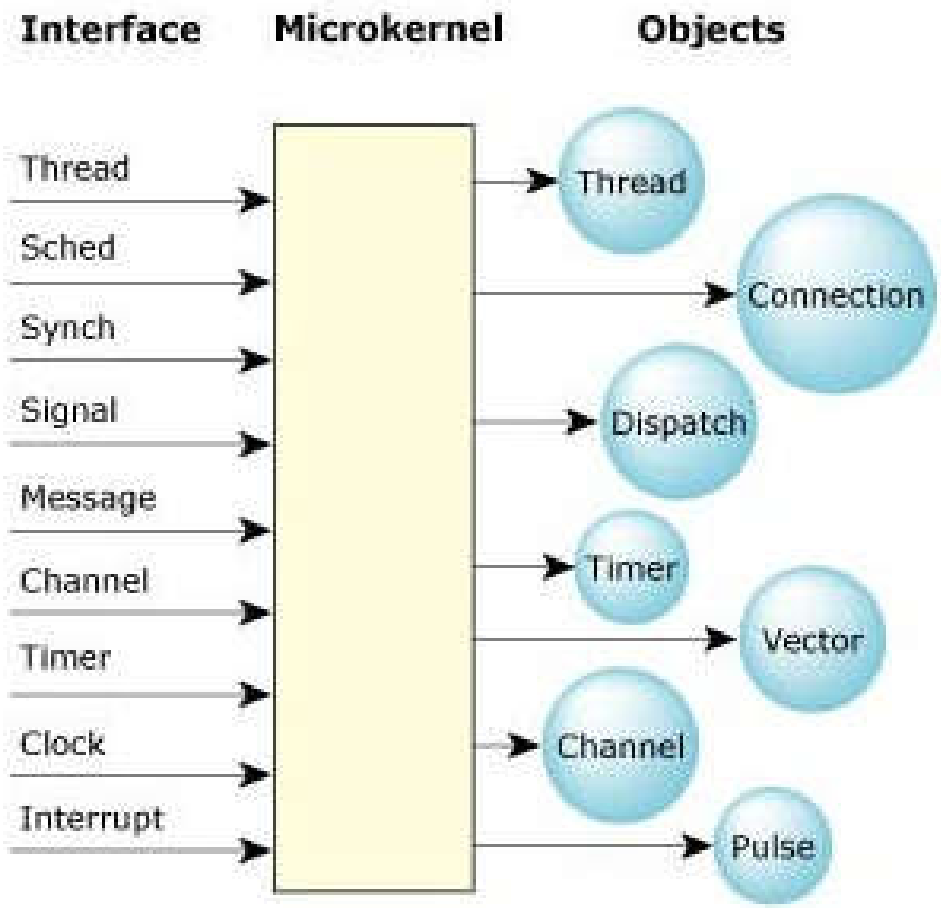


Lsass = local security authentication server
POSIX = portable operating system interface
GDI = graphics device interface
DLL = dynamic link libraries

Colored area indicates Executive

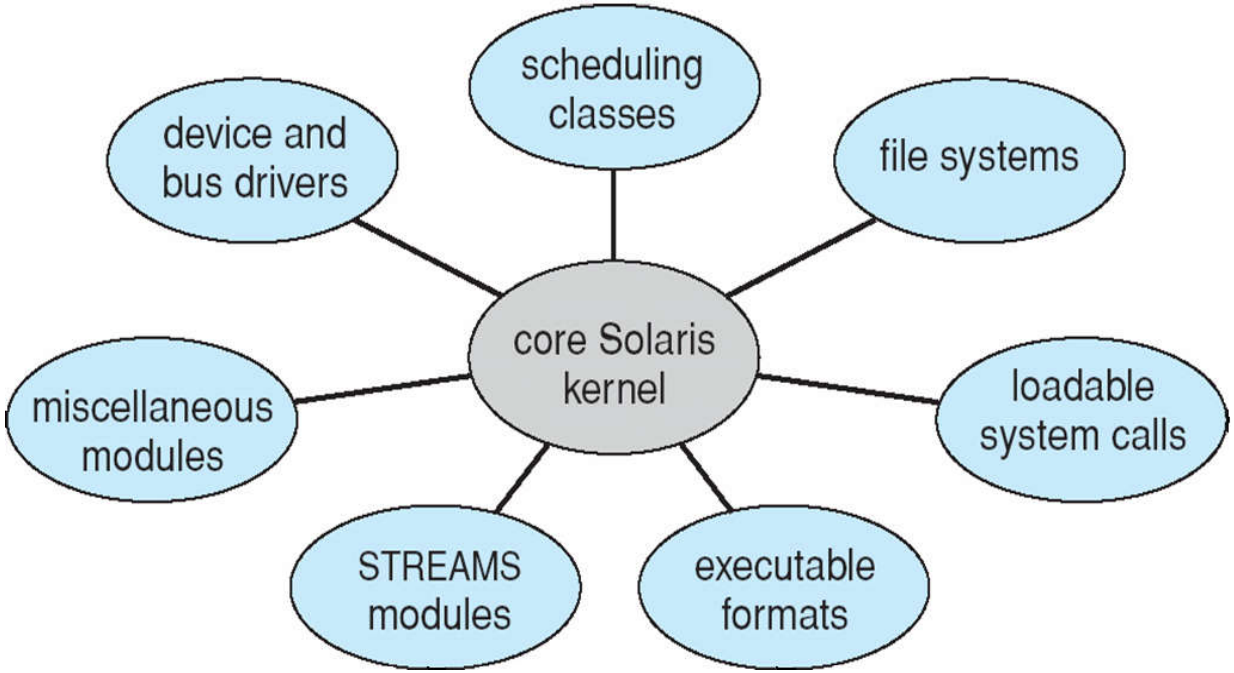
Microkernel System Structure

- The QNX Neutrino RTOS Microkernel Structure.
  - For embedded systems, supporting POSIX.



Loadable Kernel Modules (LKMs)

- Most modern operating systems implement loadable kernel modules:
  - 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.
- overall, similar to layers but with more flexibility
- Example: UNIX, LINUX, macOS, Solaris Modular Approach, and Windows

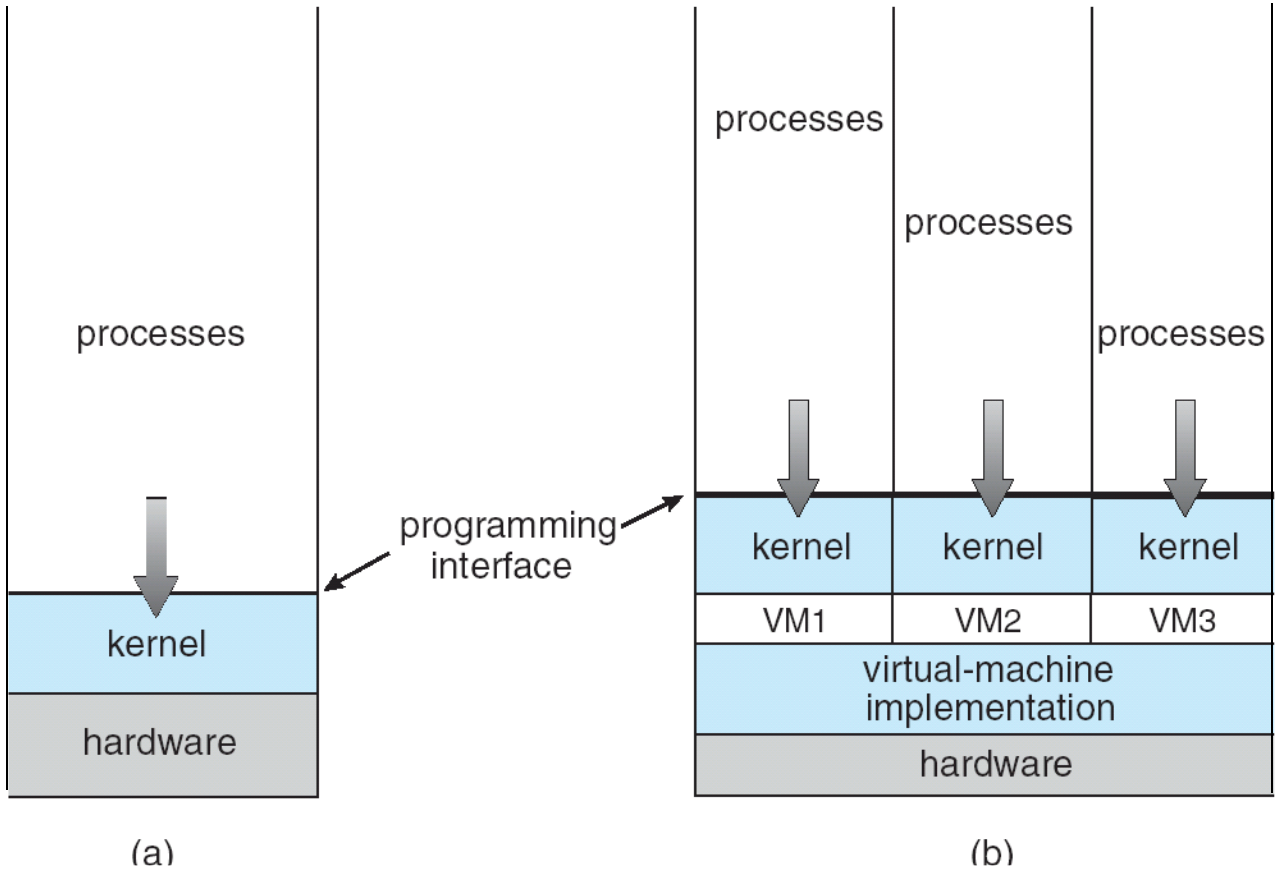


Virtual Machines

- First appeared commercially in IBM mainframes in 1972, a Virtual Machine (VM) takes the *layered* and *microkernel* approach to its logical conclusion.
- VM provides an interface identical to the underlying bare hardware.
  - It treats hardware and the operating system kernel as though they were all hardware.
- The operating system host creates the illusion that a process has its own processor and (virtual) memory.
  - Each guest provided with a (virtual) copy of underlying computer.
- The resources of the physical computer are shared to create the virtual machines:
  - CPU scheduling can create the appearance that users have their own processor.
  - SPOOLing and a file system can provide virtual card readers and virtual line printers.
  - A normal user time-sharing terminal serves as the virtual machine operator's console.

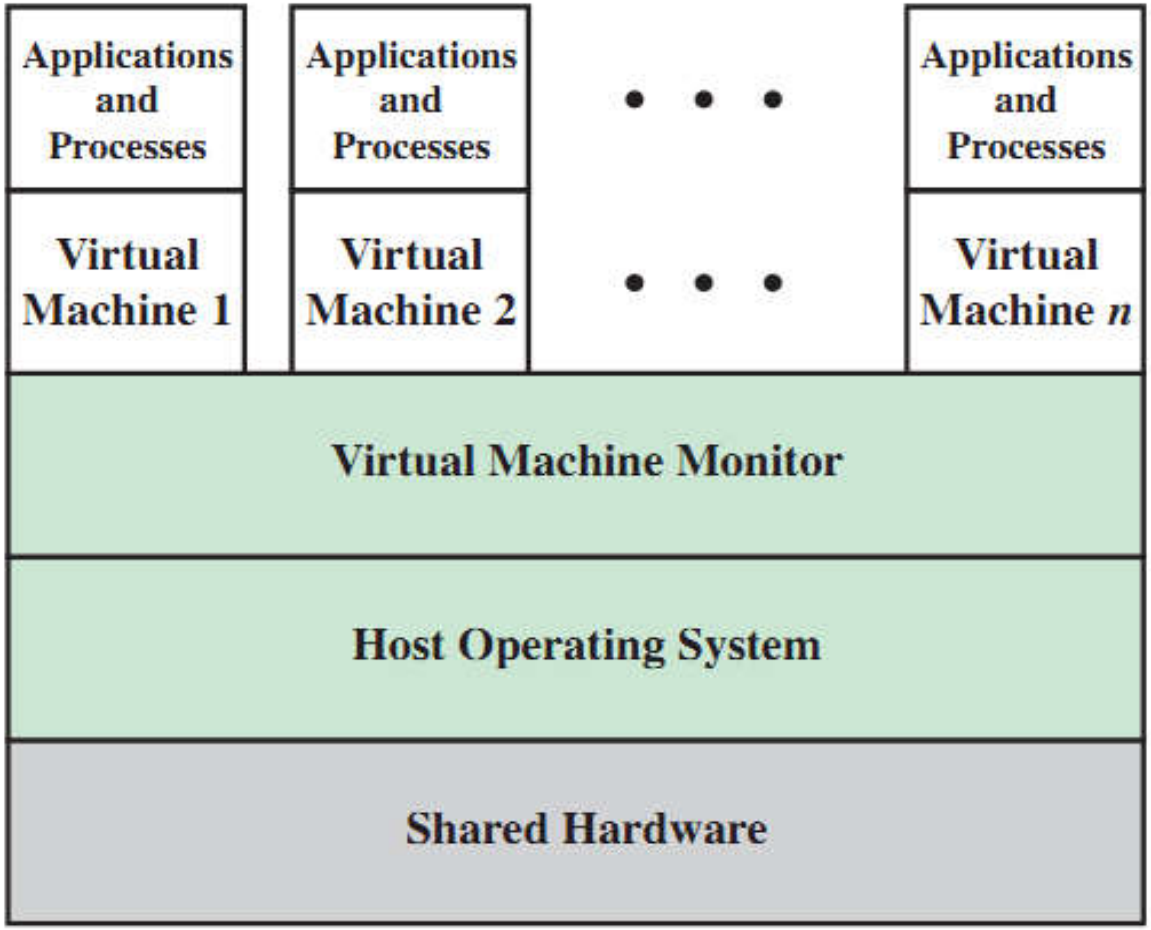
Virtual Machines

- VM Implementation on Bare Machine.
  - A computer running (a) a single operating system and (b) three virtual machines.



Virtual Machines

- VM Implementation on Host Machine.



Virtual Machines

- Advantages/Disadvantages of VMs
  - Fundamentally, multiple protected execution environments can share the same hardware.
    - complete protection of system resources since each VM is isolated from all others. This isolation permits no direct sharing of resources
    - commutate with each other or with other physical systems via networking
  - Useful for development and testing
    - System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.
  - The VM concept is difficult to implement due to the effort required to provide an exact duplicate to the underlying machine.

Virtual Machines

- Virtual appliances
  - Virtual appliances (虚拟设备) are an evolution of software delivery enabled through virtualization. They are fueling the delivery of efficient and optimized applications and also the quickly growing wave of cloud computing.
  - The OVF (Open Virtualization Format, previously called the Open Virtual Machine Format) is a specification for a portable (meaning hypervisor-neutral) distribution method for VMs. With OVF, virtual appliances can be securely packaged and distributed in an efficient way.

Virtual Machines

- Emulation vs. Virtualization
  - Emulation
    - When source CPU type is different from target type.
      - e.g., PowerPC to Intel x86.
    - Generally the slowest method.
    - Interpretation needed when native code is not generated from compiling.
  - Virtualization
    - OS host is natively compiled for CPU, the same with running guest operating systems.
      - Consider VMware running WinXP guests, each running applications, all on native WinXP host OS.
    - VMM (Virtual Machine Manager) provides virtualization services.

Virtual Machines

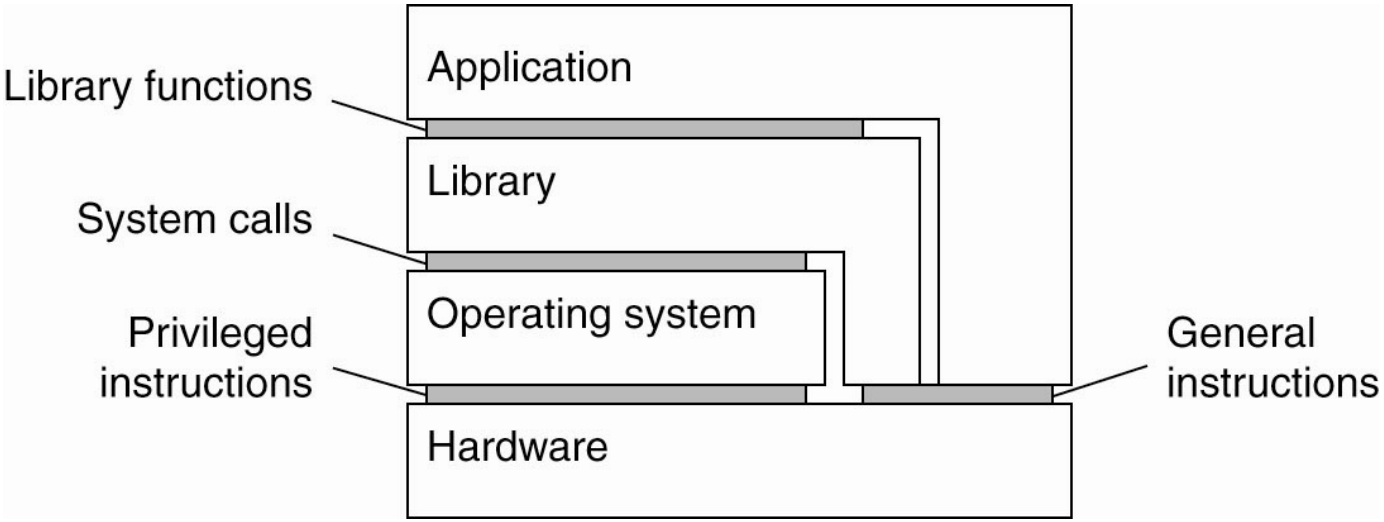
- Virtualization Examples
  - Use cases involve laptops and desktops running multiple operating systems for exploration or compatibility:
    - Apple laptop running Mac OS X host, Windows as a guest.
    - Developing apps for multiple OSes without having multiple systems.
    - QA testing applications without having multiple systems.
    - Executing and managing compute environments within data centers.
  - VMM can run natively, so they are also the host.
    - There is no general purpose host.
      - VMware ESX
      - Citrix XenServer.

Virtual Machines

- Architectures of Virtual Machines
  - There are interfaces at different levels.
  - An interface between the hardware and software, consisting of machine instructions that can be invoked by any program.
  - An interface between the hardware and software, consisting of machine instructions that can be invoked only by privileged programs, such as an operating system.
  - An interface consisting of system calls as offered by an operating system.
  - An interface consisting of library calls:
    - generally forming an Application Programming Interface (API).
    - In many cases, the aforementioned system calls are hidden by an API.

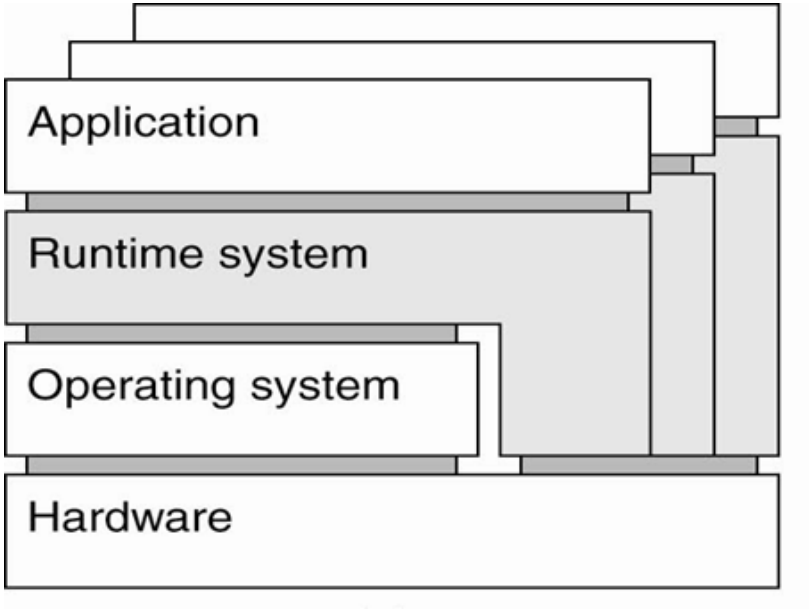
Virtual Machines

- Architectures of Virtual Machines.



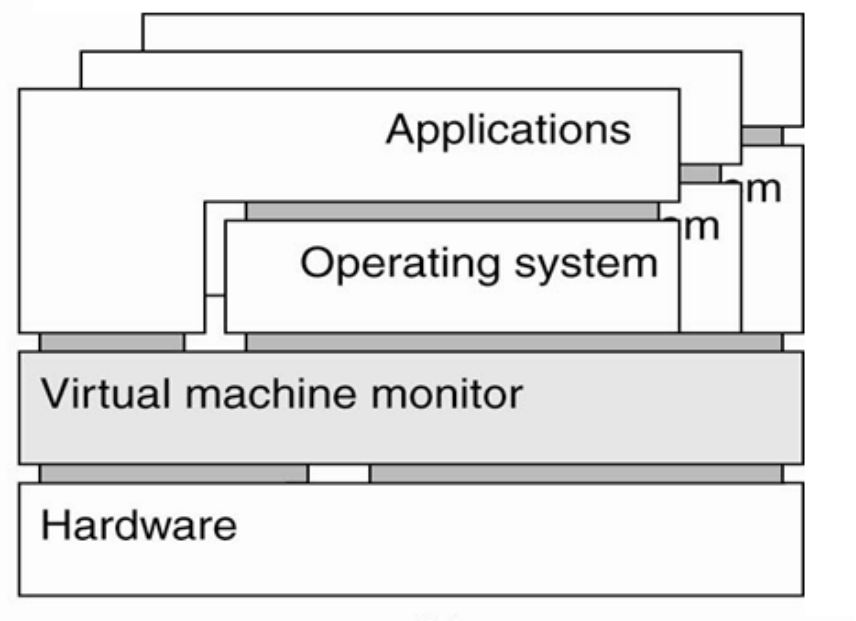
Virtual Machines

- Architectures of Virtual Machines
  - A virtual machine with multiple instances of (application, runtime) combinations.



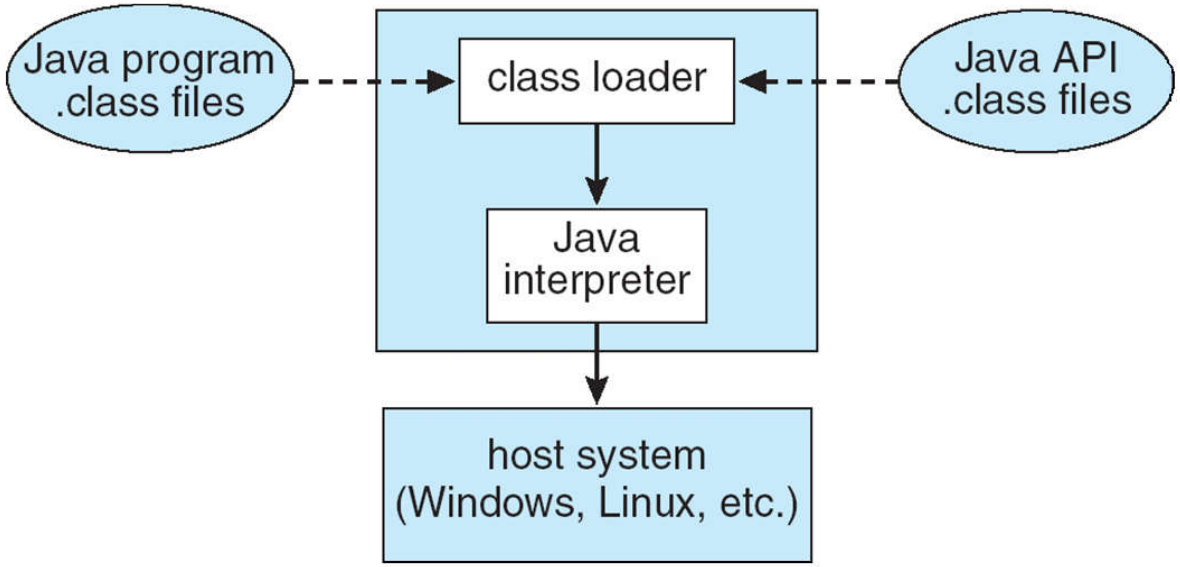
Virtual Machines

- Architectures of Virtual Machines
  - A virtual machine monitor (VMM)/Hypervisor with multiple instances of (applications, operating system) combinations.



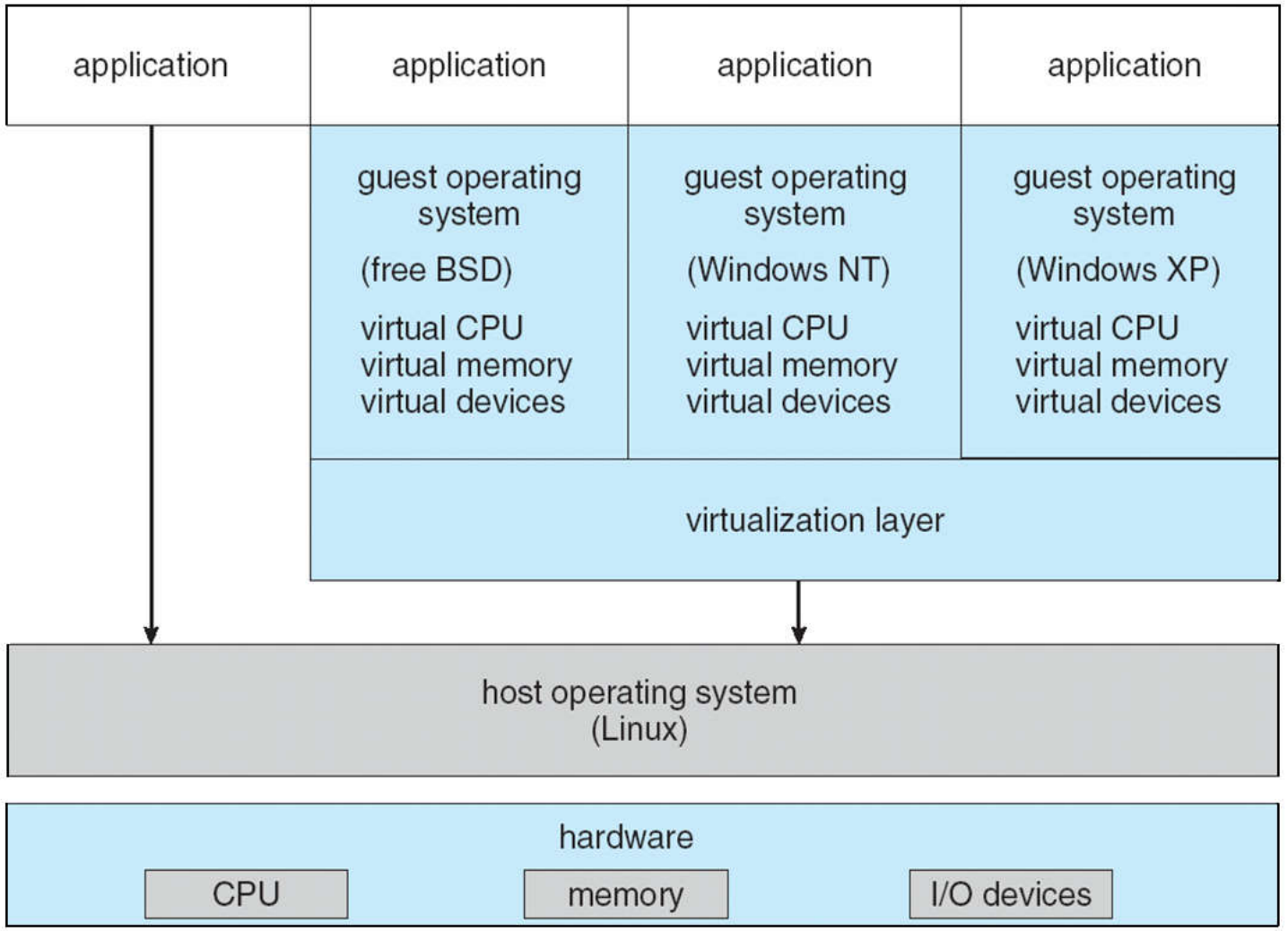
Virtual Machines

- Java Virtual Machine
  - Compiled Java programs are platform-neutral bytecodes executed by a Java Virtual Machine (JVM).
  - JVM consists of:
    - class loader
    - class verifier
    - runtime interpreter
  - Just-In-Time (JIT) compilers increase performance.



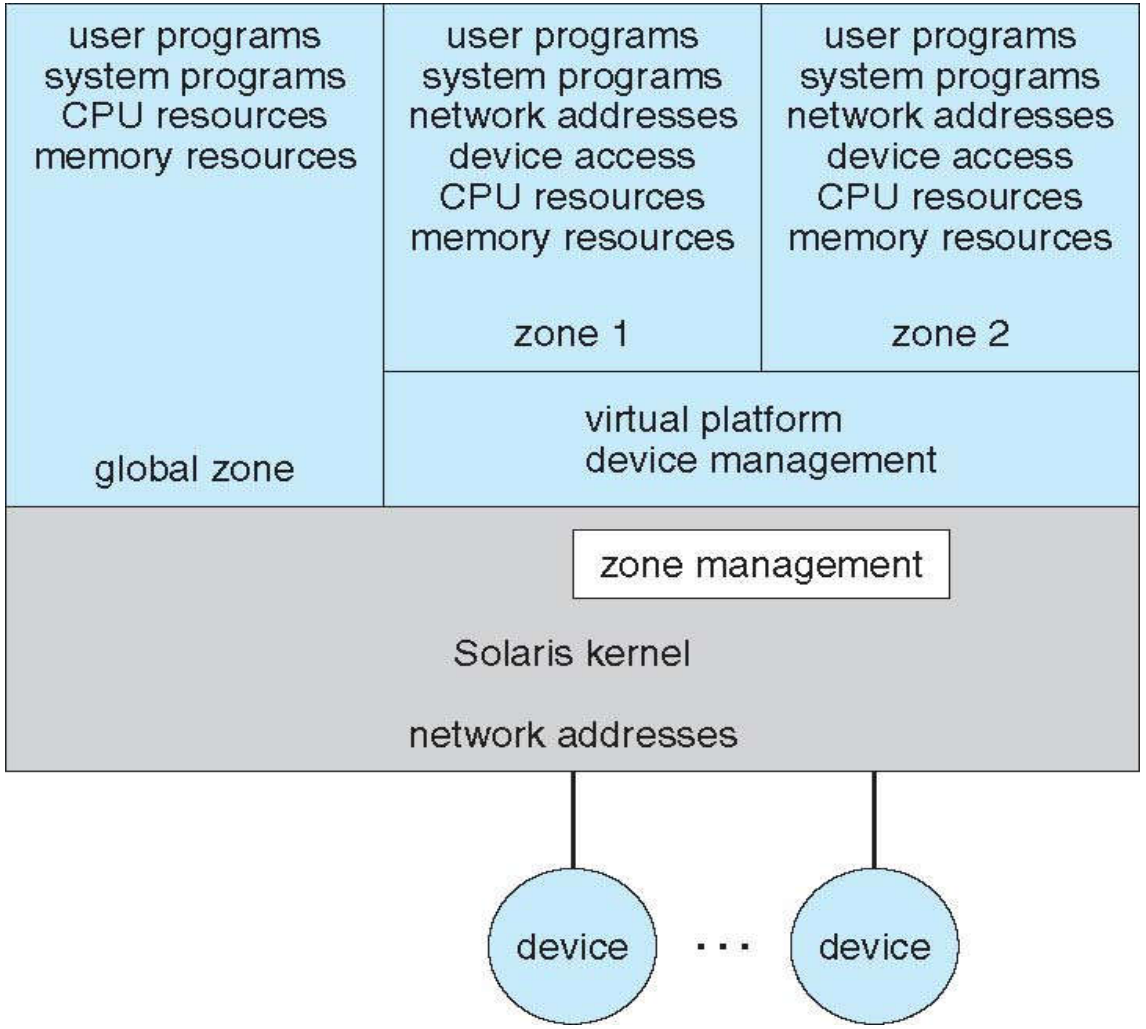
Virtual Machines

- VMware Architecture.



Virtual Machines

- Solaris 10 with Two Containers.



Inside the Linux Kernel