OPERATING SYSTEMS

Structures of Operating Systems

Operating Systems

郑贵锋 博士
中山大学计算机学院
zhenggf@mail.sysu.edu.cn
https://gitee.com/code_sysu

中山大學 SUN YAT-SEN UNIVERSITY

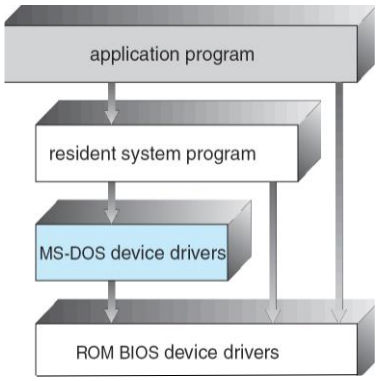
- **目录**
  - 操作系统服务
  - 通用操作系统组件
  - 系统调用和API
  - 系统程序
  - 操作系统设计与实现
  - **操作系统的结构/组织/布局**
    - Monolithic**单片（一个非结构化程序）**
    - **分层**
    - **微核**
    - **虚拟机**

Structures of Operating Systems

- 单片操作系统
  - 单片操作系统的基本结构(宏内核)
    - 应用程序调用请求的系统服务
    - 一组系统服务执行操作系统的过程/调用
    - 一组实用程序辅助系统服务
    - 内核的所有功能都放在一个静态二进制文件中，该文件在一个地址空间中运行。
  - MS-DOS
    - MS-DOS的编写目的是在最少的空间内提供最多的功能
    - 虽然MS-DOS有一些结构，但其接口和功能级别并没有很好地分开
      - 不分为模块

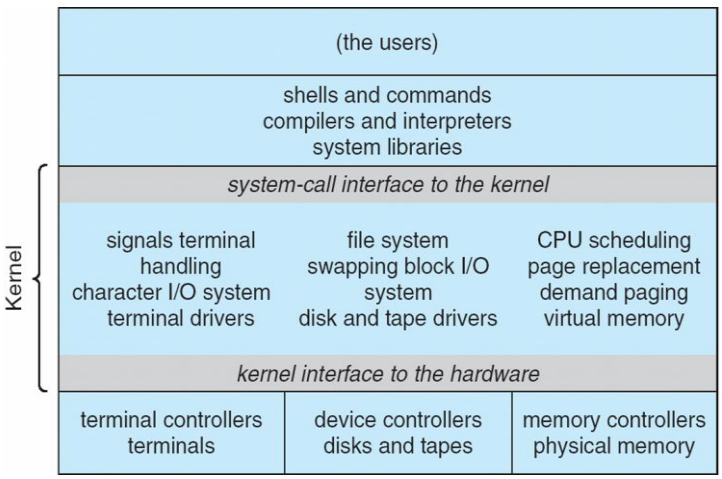
Structures of Operating Systems

- 单片操作系统
  - MS-DOS
    - MS-DOS分层结构



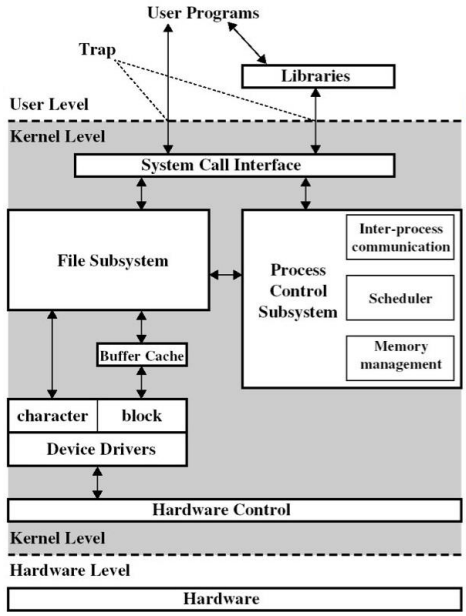
- 单片操作系统
  - UNIX
    - UNIX受到硬件功能的限制，最初的UNIX操作系统结构有限。
    - UNIX由两个可分离的部分组成：
      - 系统程序
      - 内核
        - 包括系统调用接口下方和物理硬件上方的所有内容。
        - 提供文件系统、CPU调度、内存管理等操作系统功能；一个层次的大量功能。

- 单片操作系统
  - UNIX
    - 传统的UNIX系统结构



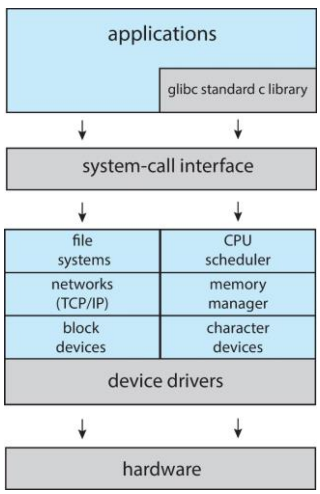
Structures of Operating Systems

- **单片操作系统**
  - UNIX
    - 传统UNIX内核



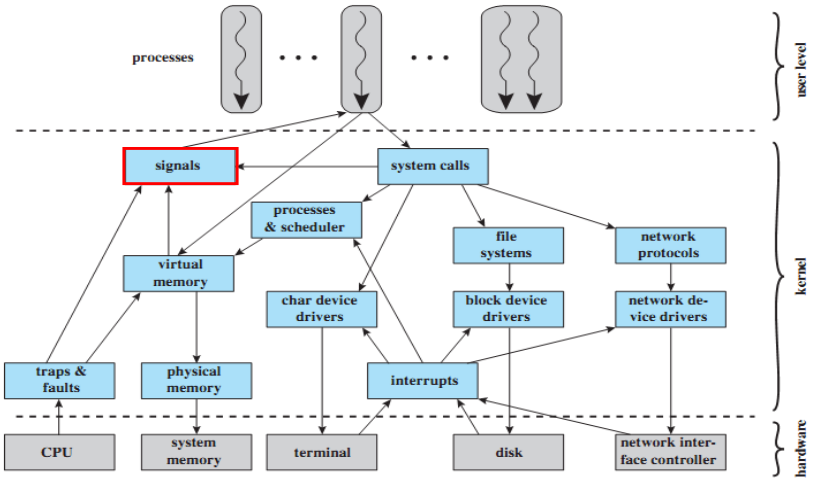
Structures of Operating Systems

- **单片操作系统**
  - Linux
    - Linux系统结构



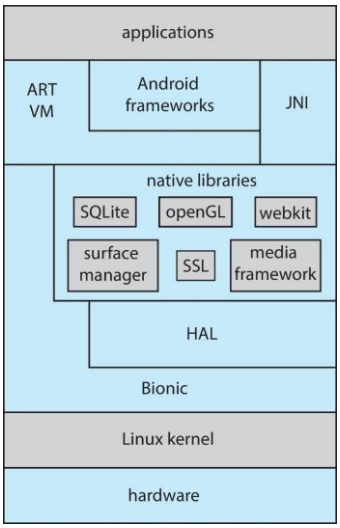
Structures of Operating Systems

- 单片操作系统
  - Linux
    - Linux内核组件



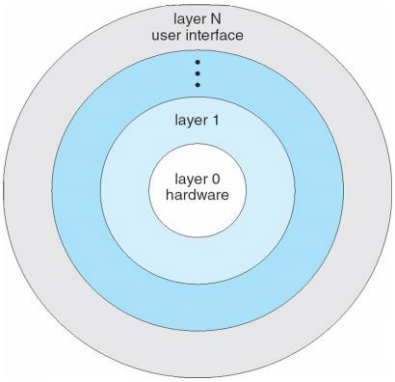
Structures of Operating Systems

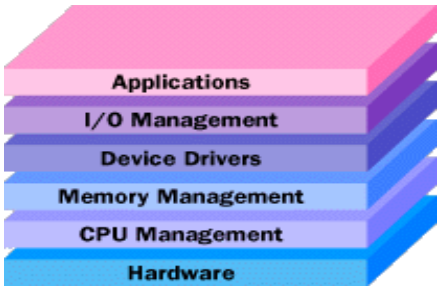
- 单片操作系统
  - 谷歌的Android系统



Structures of Operating Systems

- 分层方法
  - 操作系统分为若干层（级别），每个层构建在较低层的顶部。
  - 底层（0层）是硬件；最高层（第N层）是用户界面。
  - 在模块化的情况下，层的选择使得每个层只使用较低级别层的功能（操作）和服务。



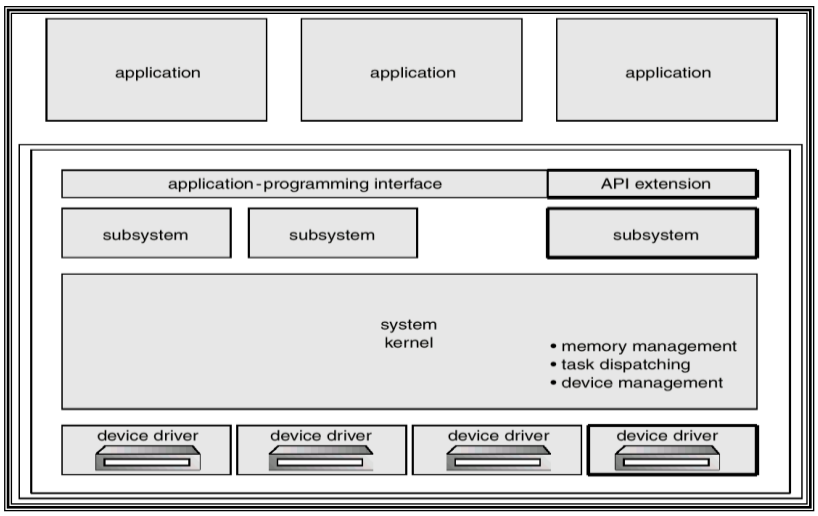


Structures of Operating Systems

- 分层方法
  - 旧版本的Windows系统层

Structures of Operating Systems

- 分层方法
  - OS/2层结构

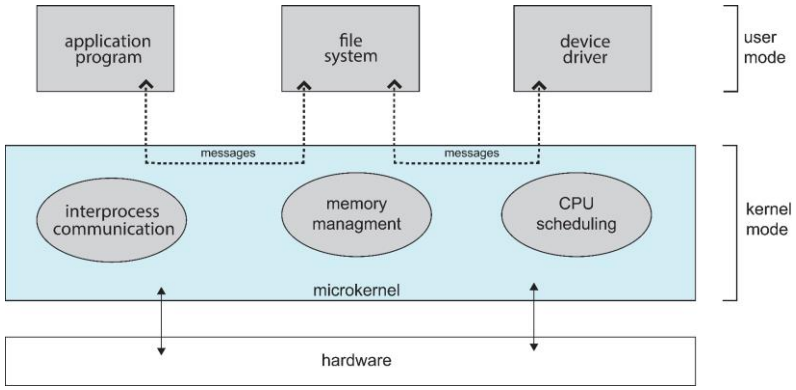


Structures of Operating Systems

- 微内核结构
  - 微内核系统将尽可能多的功能从内核转移到“用户”空间。
    - 随着UNIX的扩展，内核变得庞大且难以管理。
    - Mach 3.0操作系统（CMU，1985-1994）使用微内核方法对内核进行模块化。
    - 结果是一个更小的内核，内核中只保留了几个基本函数：
      - 基本内存管理（地址空间）
      - I/O和中断管理
      - 进程间通信（IPC）
      - 基本调度
    - 其他操作系统服务由以用户模式运行的进程（垂直服务器）提供：
      - 设备驱动程序、文件系统、虚拟内存...
    - 使用消息传递在用户模块之间进行通信。

微内核结构

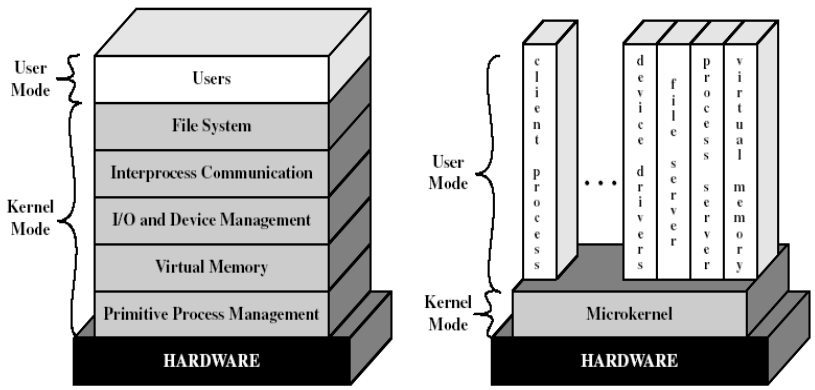
- 使用进程之间的消息交换替换服务调用会导致性能开销。



典型微内核的体系结构

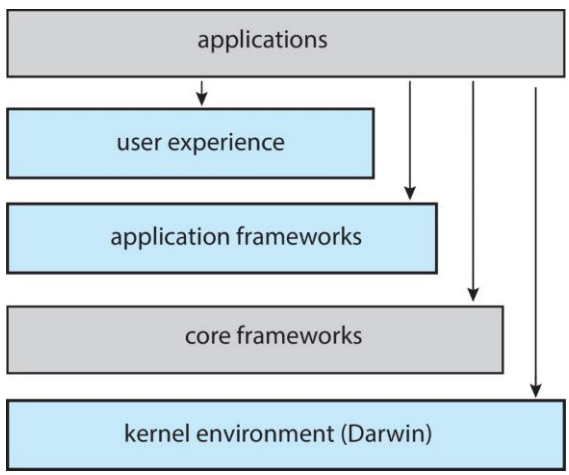
微内核结构

- 分层与微核架构

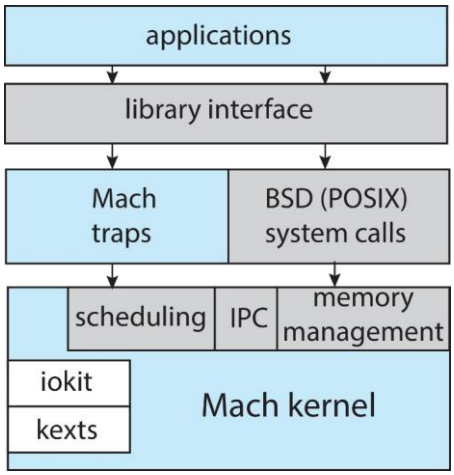


- 微内核结构
  - 微核组织的好处
    - 可扩展性/可靠性
      - 更容易扩展微内核
      - 更容易将操作系统移植到新的硬件体系结构
      - 更可靠（内核模式下运行的代码更少）
      - 更安全
      - 小微内核可以进行严格的测试
    - 可移植
      - 移植只需更改微核，而不用更改其他服务
    - 分布式系统支持
      - 可在不知道目标机器是什么的情况下发送消息
    - 面向对象操作系统
      - 组件是具有明确定义的接口的对象，可相互连接形成软件

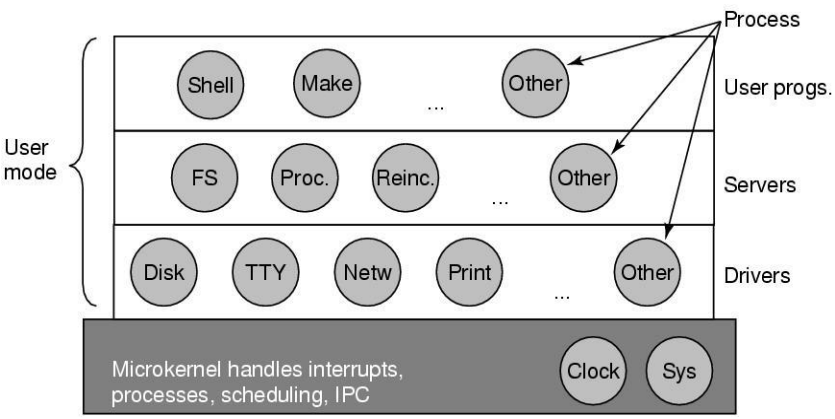
- 微内核结构
  - MacOS系统结构
    - 苹果macOS和iOS操作系统的架构



- **微内核结构**
  - MacOS系统结构
    - Darwin的结构

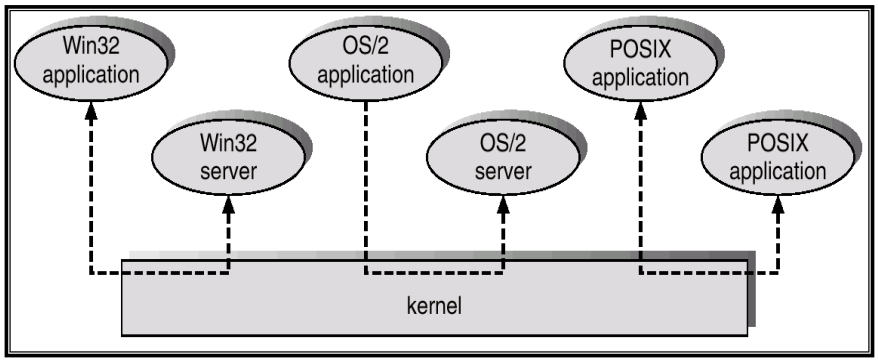


- **微内核结构**
  - MINIX 3结构
    - Linus Torvalds vs. Andrew Tanenbaum：关于单片和微核的争论



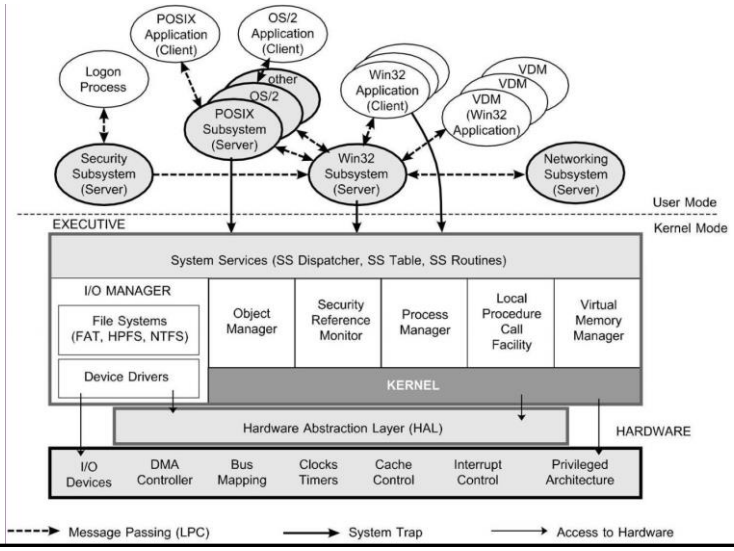
Structures of Operating Systems

- 微内核结构
  - Windows NT客户端/服务器结构。



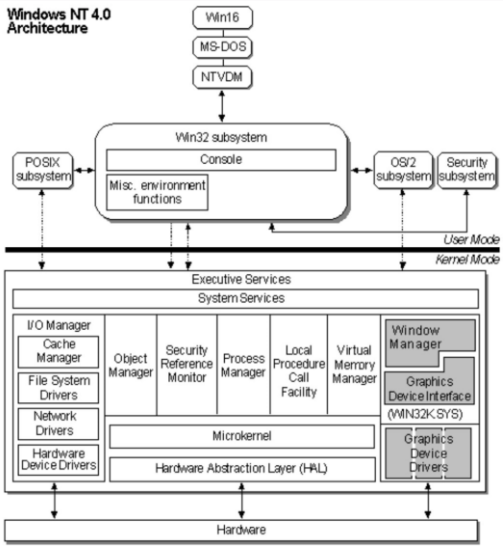
Structures of Operating Systems

- 微内核结构
  - Windows NT 4.0体系结构



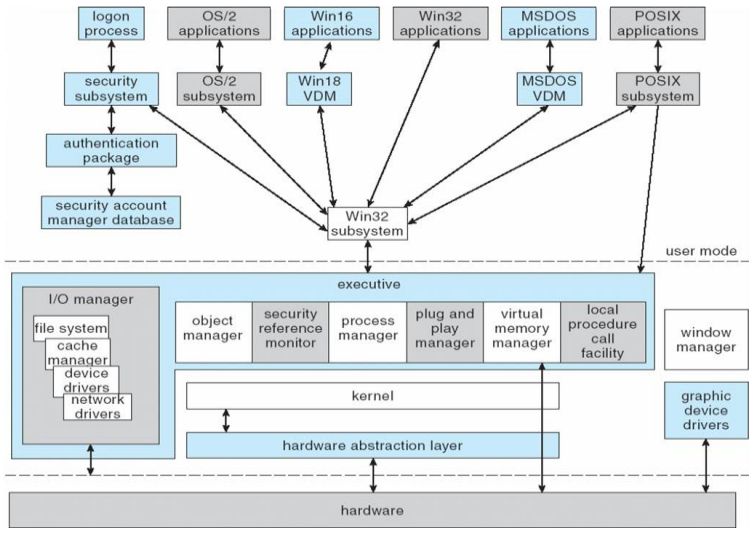
Structures of Operating Systems

- 微内核结构
  - Windows NT 4.0体系结构。



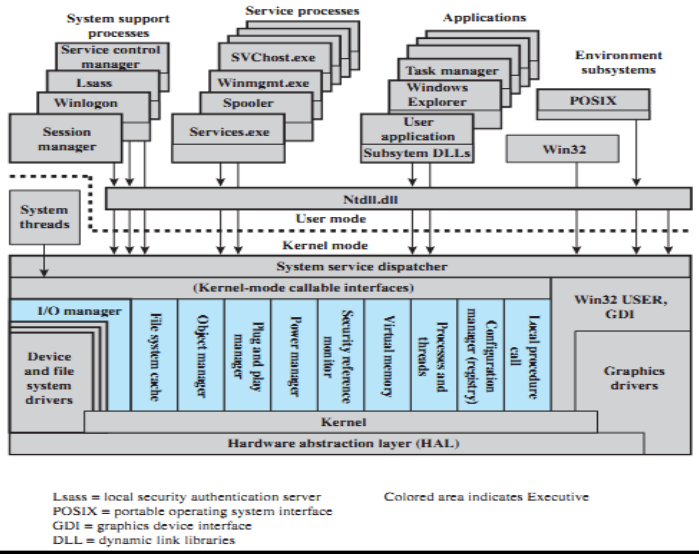
Structures of Operating Systems

- 微内核结构
  - Windows XP体系结构



Structures of Operating Systems

- 微内核结构
  - Windows 7.0体系结构

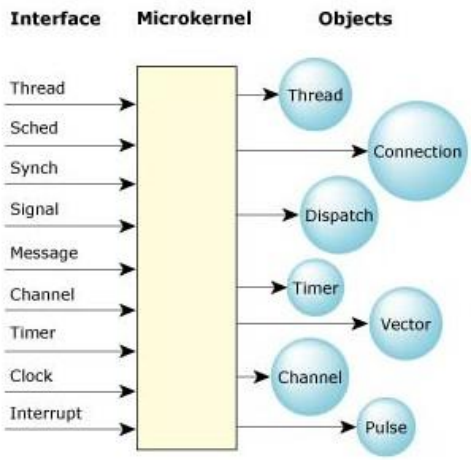


Lsass = local security authentication server
POSIX = portable operating system interface
GDI = graphics device interface
DLL = dynamic link libraries

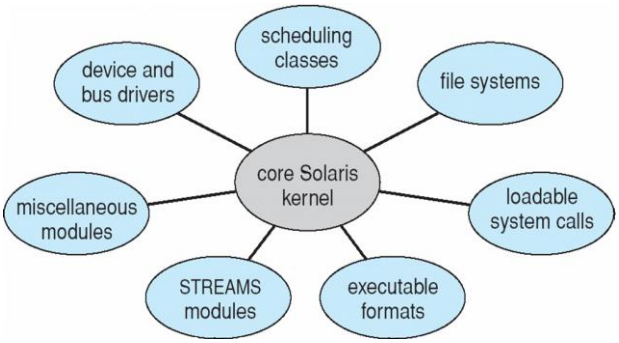
Colored area indicates Executive

Structures of Operating Systems

- 微内核结构
  - QNX Neutrino RTOS微内核结构。
    - 面向嵌入式系统，支持POSIX



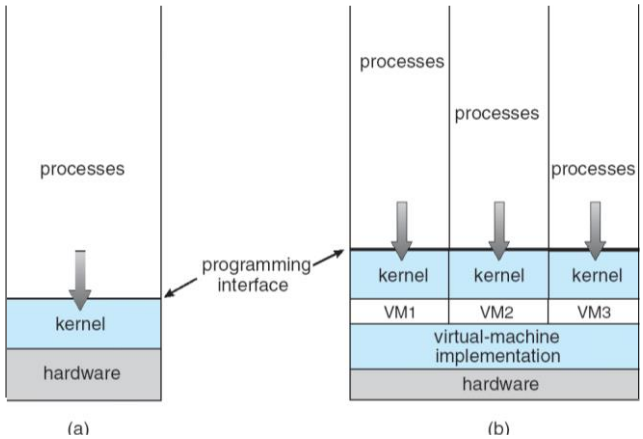
- 可加载内核模块（LKM）
  - 大多数现代操作系统都实现可加载的内核模块：
    - 使用面向对象的方法
    - 每个核心组件都是独立的
    - 模块间通过已知的接口相互通信
    - 模块都可以根据需要在内核中加载
  - 总体而言，与图层类似，但具有更大的灵活性
  - 示例：
    - UNIX
    - LINUX
    - macOS
    - Solaris
    - Windows



- 虚拟机
  - 虚拟机（VM）于1972年首次在IBM大型机上商业化出现，其逻辑结论是采用分层和微核方法。
  - VM提供与底层裸硬件相同的接口。
    - 它将硬件和操作系统内核视为所有硬件。
  - 操作系统主机产生了一种错觉，使进程认为具有自己的处理器和（虚拟）内存。
    - 每个客机都提供了底层计算机的（虚拟）副本。
  - 共享物理计算机的资源以创建虚拟机：
    - CPU调度可以创建用户拥有自己处理器的假像。
    - 假脱机和文件系统可以提供虚拟读卡器和虚拟行打印机。
    - 普通的用户分时终端充当虚拟机操作员的控制台。

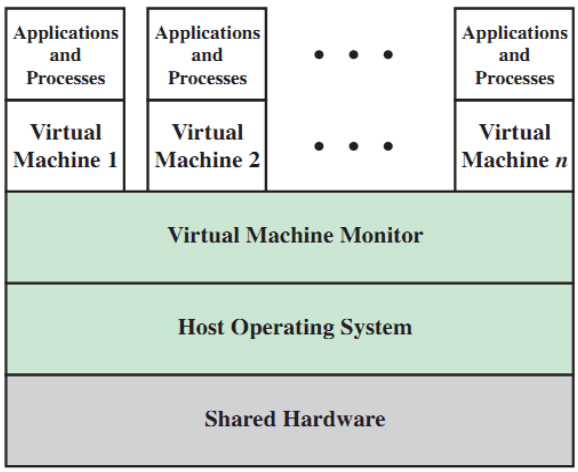
Structures of Operating Systems

- 虚拟机
  - 虚拟机在裸机上的实现。
    - 运行（a）单个操作系统和（b）三个虚拟机的计算机。



Structures of Operating Systems

- 虚拟机
  - 主机上的VM实现



虚拟机

- 虚拟机的优缺点
  - 基本上，多个受保护的执行环境可以共享同一硬件。
    - 完全保护系统资源，因为每个VM都与所有其他VM隔离。这种隔离不允许直接共享资源
    - 通过网络相互交换或与其他物理系统交换
  - 对开发和测试有用
    - 系统开发在虚拟机上完成，而不是在物理机上完成，因此不会中断正常的系统操作。
  - VM概念很难实现，因为需要为底层机器提供精确的副本。

虚拟机

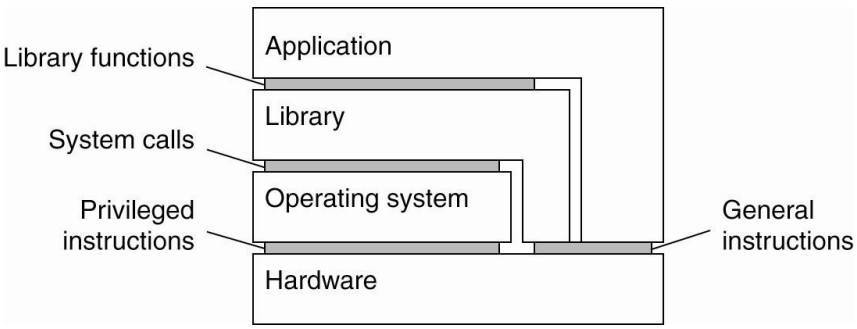
- 虚拟设备
  - 虚拟设备是通过虚拟化实现的软件交付的变革。它们正在推动高效优化应用程序的交付，也在推动快速增长的云计算浪潮。
  - OVF（开放虚拟化格式，前称开放虚拟机格式）是虚拟机的可移植（即与虚拟机监控程序无关）分发方法的规范。有了OVF，虚拟设备可以以高效的方式安全地打包和分发。

- 虚拟机
  - 仿真与虚拟化
    - 仿真Emulation
      - 当源CPU类型与目标类型不同时。
        - 例如，从PowerPC到Intel x86
      - 通常是最慢的方法
      - 编译时未生成本机代码，运行时需要进行解释
    - 虚拟化
      - 操作系统主机是为CPU本机编译的，客机操作系统也一样
        - 考虑VMware运行的WinXP客机里，每个应用程序都运行在原生WinXP主机OS上
      - VMM（虚拟机管理器）提供虚拟化服务。

- 虚拟机
  - 虚拟化示例
    - 用例涉及运行多个操作系统的笔记本电脑和台式机，以实现探索性或兼容性：
      - 运行Mac OS X主机、Windows客机的苹果笔记本电脑。
      - 在没有多个系统的情况下为多个操作系统开发应用程序。
      - 没有多个系统的QA测试应用程序。
      - 在数据中心内执行和管理计算环境。
    - VMM可以本机运行，因此它们也是主机。
      - 没有通用主机
        - VMware ESX
        - Citrix XenServer

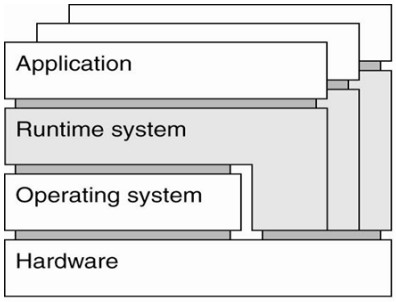
- 虚拟机
  - 虚拟机体系结构
    - 有不同级别的接口
    - 硬件和软件之间的接口，由可由任何程序调用的机器指令组成
    - 硬件和软件之间的接口，由只能由特权程序（如操作系统）调用的机器指令组成。
    - 由操作系统提供的系统调用组成的接口。
    - 由库调用组成的接口：
      - 通常形成应用程序编程接口（API）。
      - 在许多情况下，上述系统调用被API隐藏。

- 虚拟机
  - 虚拟机的体系结构。



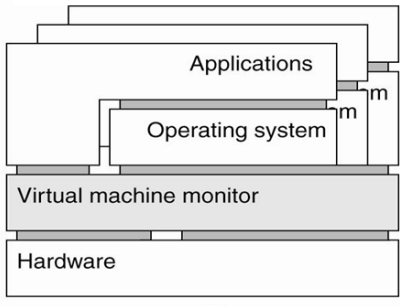
Structures of Operating Systems

- 虚拟机
  - 虚拟机体系结构
    - 具有多个（应用程序、运行时）组合实例的虚拟机。



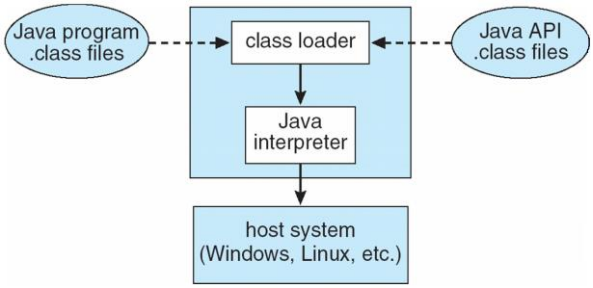
Structures of Operating Systems

- 虚拟机
  - 虚拟机体系结构
    - 具有（应用程序、操作系统）组合的多个实例的虚拟机监视器（VMM）/虚拟机监控程序。



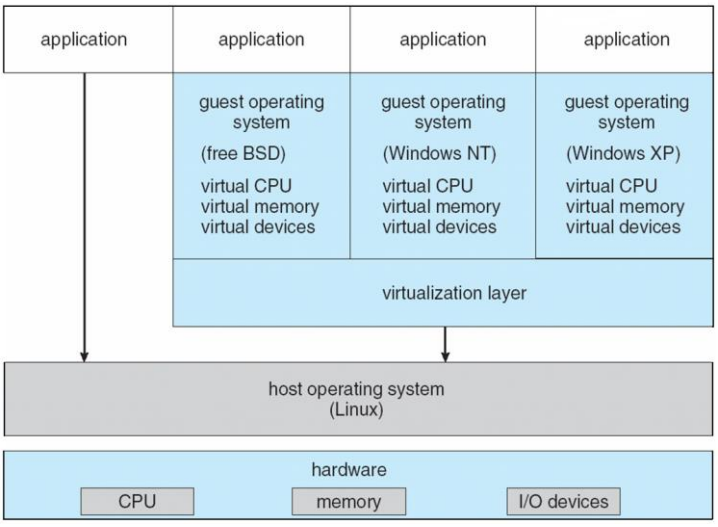
虚拟机

- Java虚拟机
  - 编译的Java程序是由Java虚拟机（JVM）执行的平台无关字节码
  - JVM包括：
    - 类加载器
    - 类验证器
    - 运行时解释器
  - 即时（JIT）编译器提高了性能。



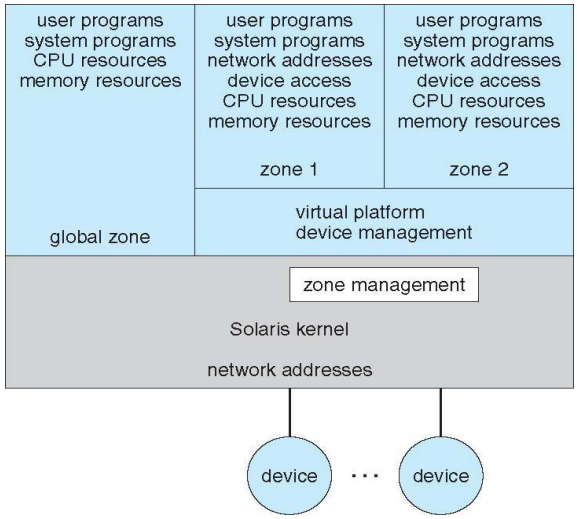
虚拟机

- VMware体系结构



Structures of Operating Systems

- 虚拟机
  - Solaris 10，带有两个容器。



Structures of Operating Systems

- Linux内核内部