OS Structures

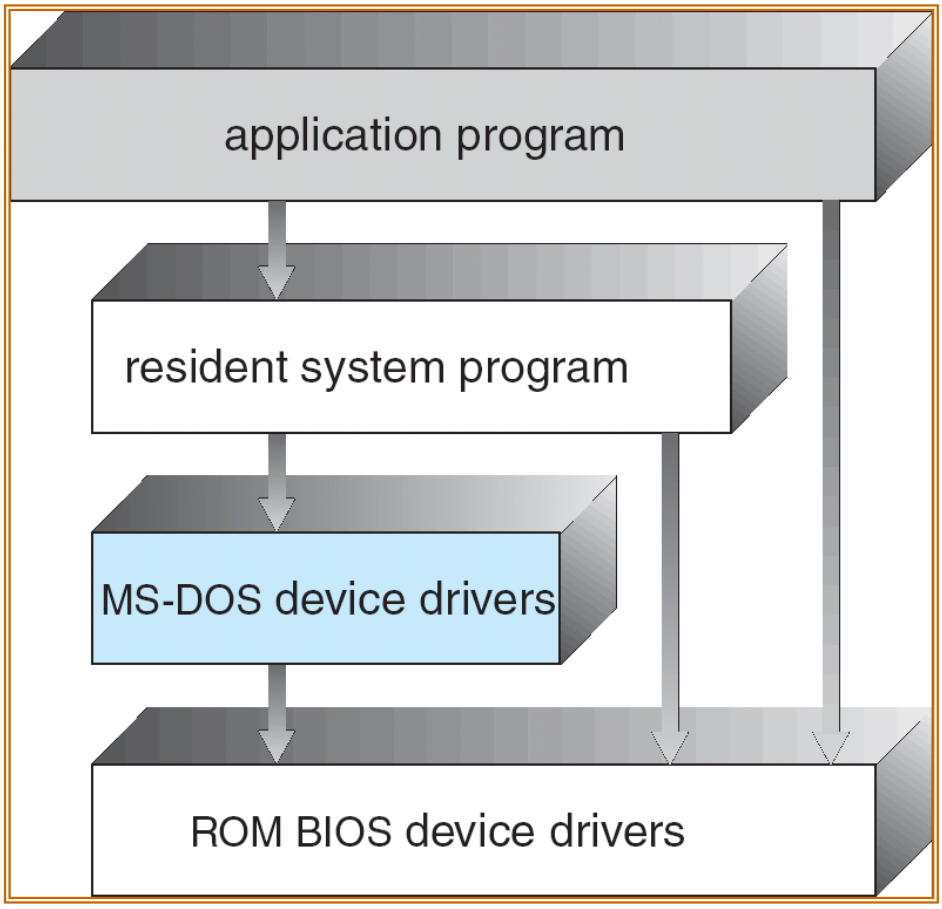
Simple Structure

- Do not have well defined structure
- Such OS started as small, simple and limited OS
- e.g. MS-DOS, UNIX

MS-DOS Structure

- 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

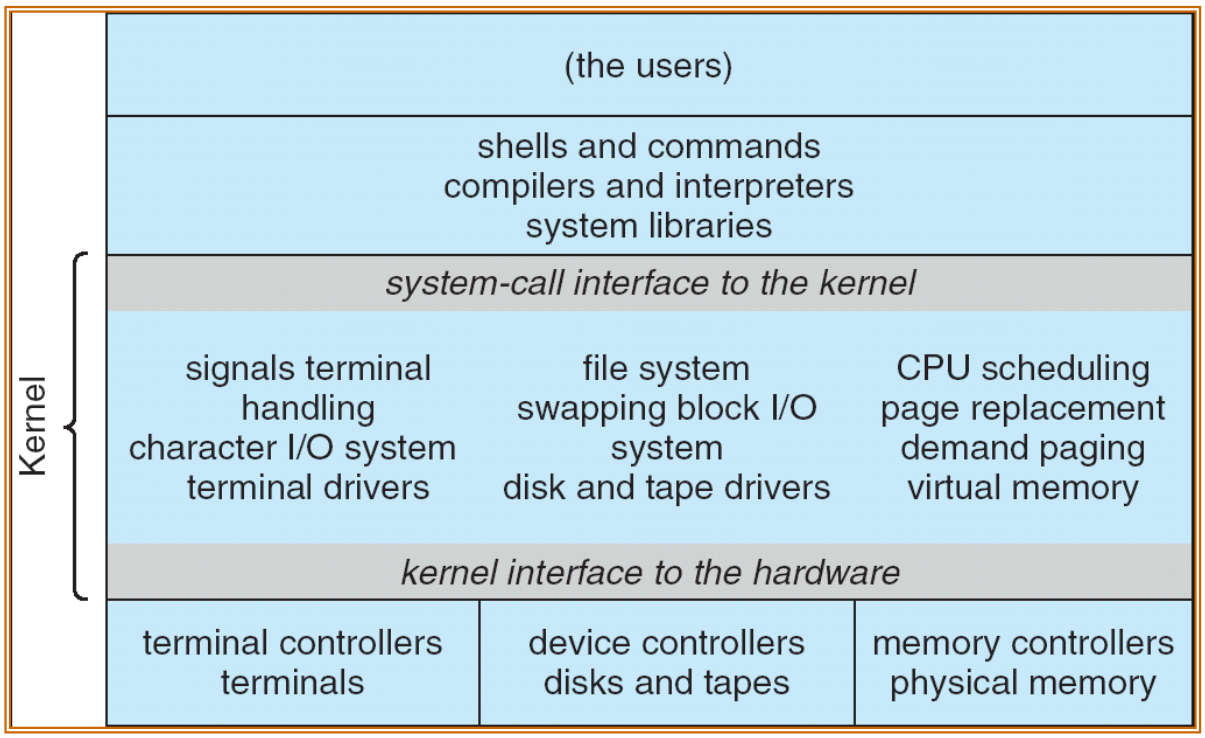
MS-DOS Layer Structure



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 & maintain.

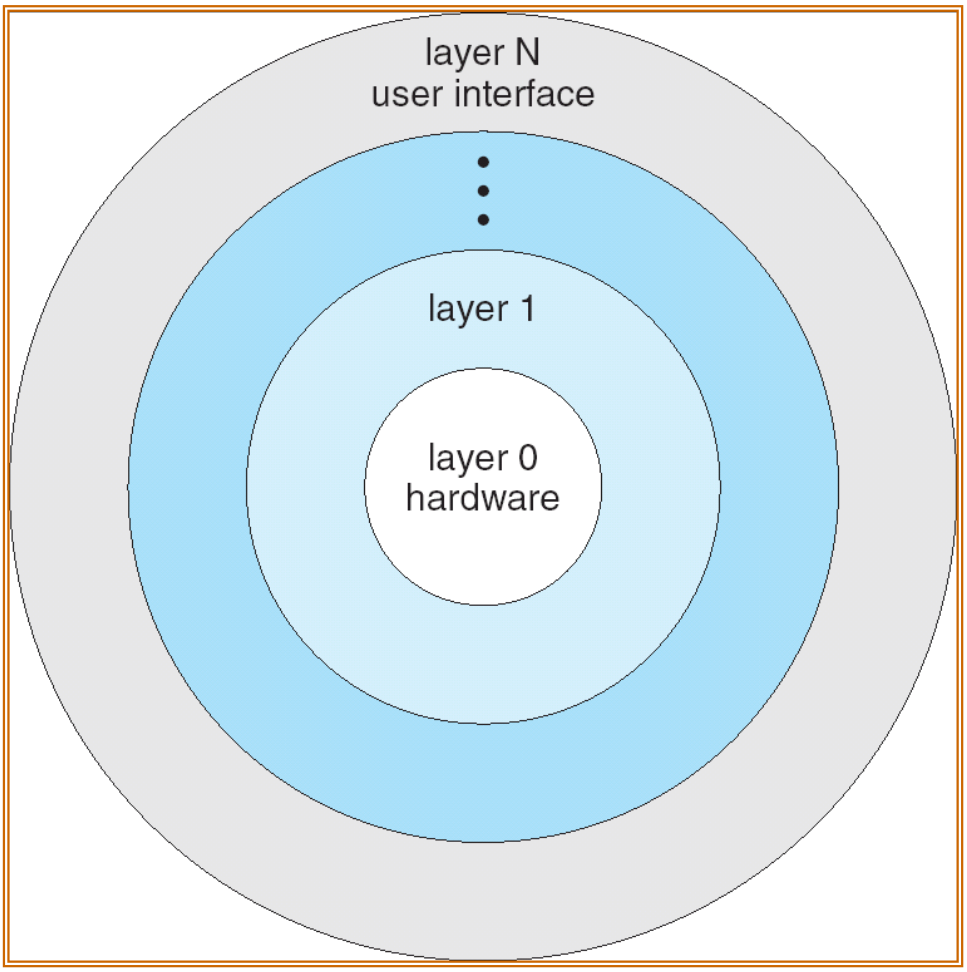
UNIX System Structure



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 Operating System



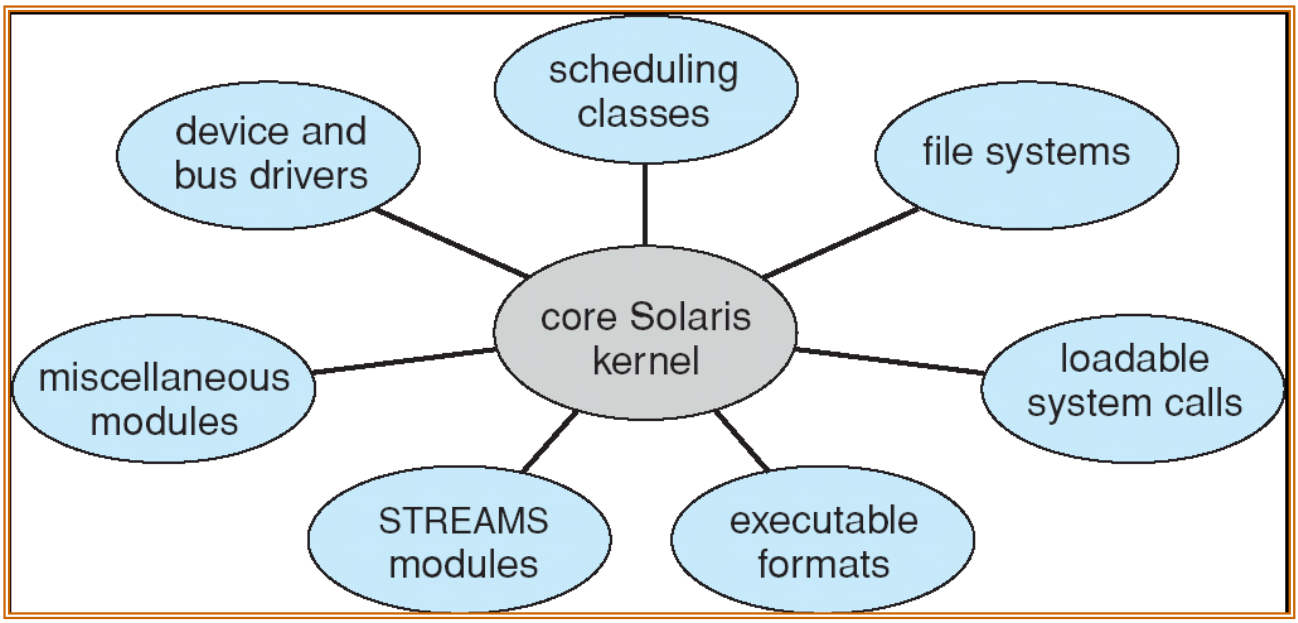
Microkernel System Structure

- Structures the OS by removing all non-essential components from the kernel and implementing them as system & user level programs. Moves as much from the kernel into "*user*" space.
- Result is the smaller kernel
- 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)

Modules

- Most modern operating systems implement kernel modules
  - Uses object-oriented approach
  - Each core component is separate
  - Kernel has a set of core components & dynamically links in additional services either during boot time or run time.
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexibility
- e.g. Solaris, LINUX, Mac OS X

Solaris Modular Approach



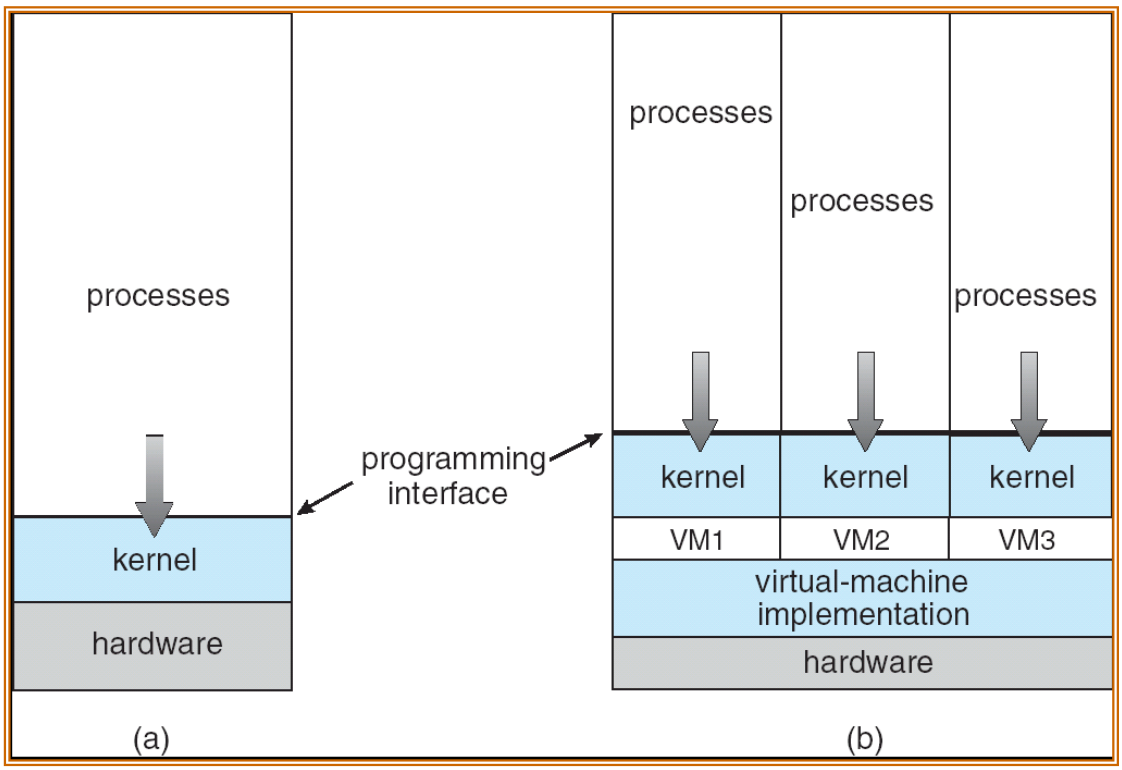
Virtual Machines

- Fundamental idea is to abstract the hardware of a single computer (CPU, memory, disk drives, network interface cards) into several different execution environments; thereby creating the illusion that each separate execution environment is running its own private computer.
- A virtual machine provides an interface *identical* to the underlying bare hardware.
- Each process is provided with a (virtual) copy of the underlying computer.
- The operating system creates the illusion of multiple processes, each executing on its own processor with its own (virtual) memory.

Virtual Machines (Cont.)

- 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
- VMM- Virtual Machine Manager provides virtualization services.
- e.g. of VMM: VMware ES X, Citrix XenServer, VMware Player (free on Windows platform), VirtualBox (open-source & free on many platforms)

Virtual Machines (Cont.)

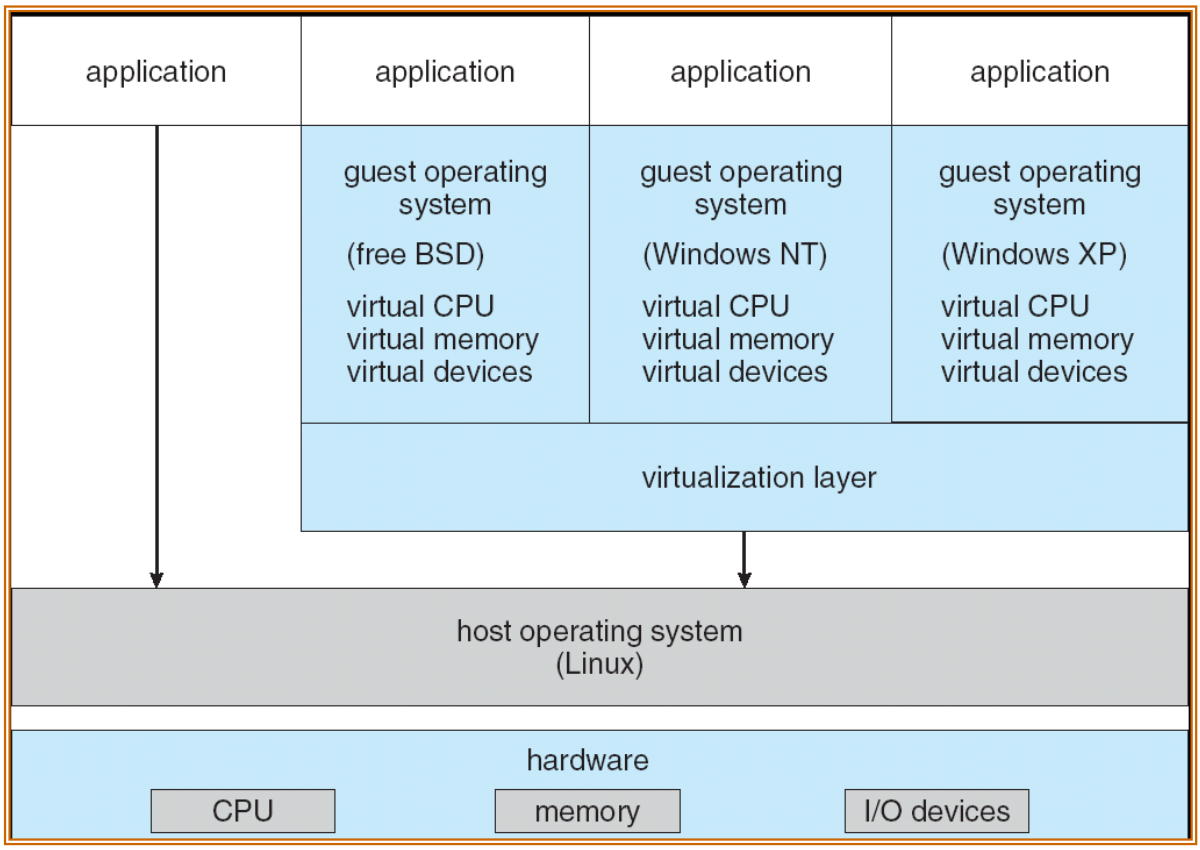


(a) Nonvirtual machine (b) virtual machine

Virtual Machines (Cont.)

- The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
- A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.

VMware Architecture



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

The Java Virtual Machine