System calls…..

- C-program->POSIX call
- POSIX->OS level call (Linux has ~322 syscalls)
  - Store the system call number in some standard location
  - Store parameters in some standard location
  - Trap to kernel
    - Use the system call number to find the correct function
    - Use the stored parameters
    - Store return values in the the block
    - Return
  - Return POSIX complaint error status

System Programs

- Provide a convenient environment for program development and execution. Some of them are simply user interfaces to system calls; others are considerably more complex
  - File management - Create, delete, copy, edit, rename, print, dump, list, and generally manipulate files and directories
  - Programming-language support - Compilers, assemblers, debuggers and interpreters sometimes provided
  - Program loading and execution- Absolute loaders, relocatable loaders, linkage editors, and overlay-loaders, debugging systems for higher-level and machine language
  - Communications - chat, web browsing, email, remote login, file transfers
  - Status information - system info such as date, time, amount of available memory, disk space, number of users

Operating System Design and Implementation

- Design and Implementation of OS affected by choice of hardware, type of system
- *User* goals and *System* goals
  - User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast
  - System goals – operating system should be easy to design, implement, and **maintain (portable?)**, as well as flexible, reliable, error-free, and efficient
- Important principle to separate

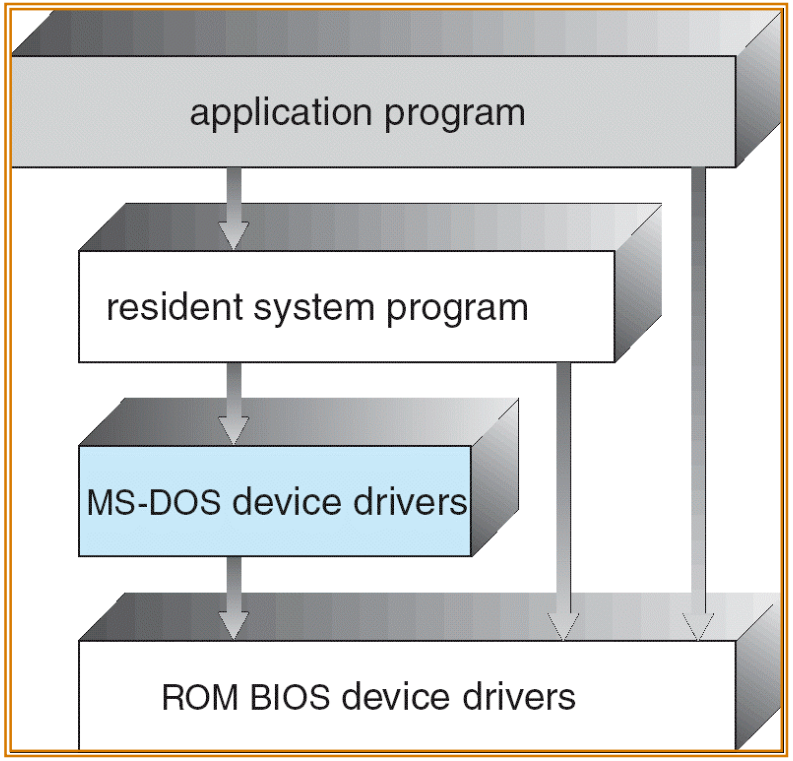
  Policy: What will be done?
  Mechanism: How to do it?
  - The separation of policy from mechanism is a very important principle, it allows maximum flexibility if policy decisions are to be changed later

OS Structure

- Simple
- Layered
- Microkernel
- Modular

Simple Structure

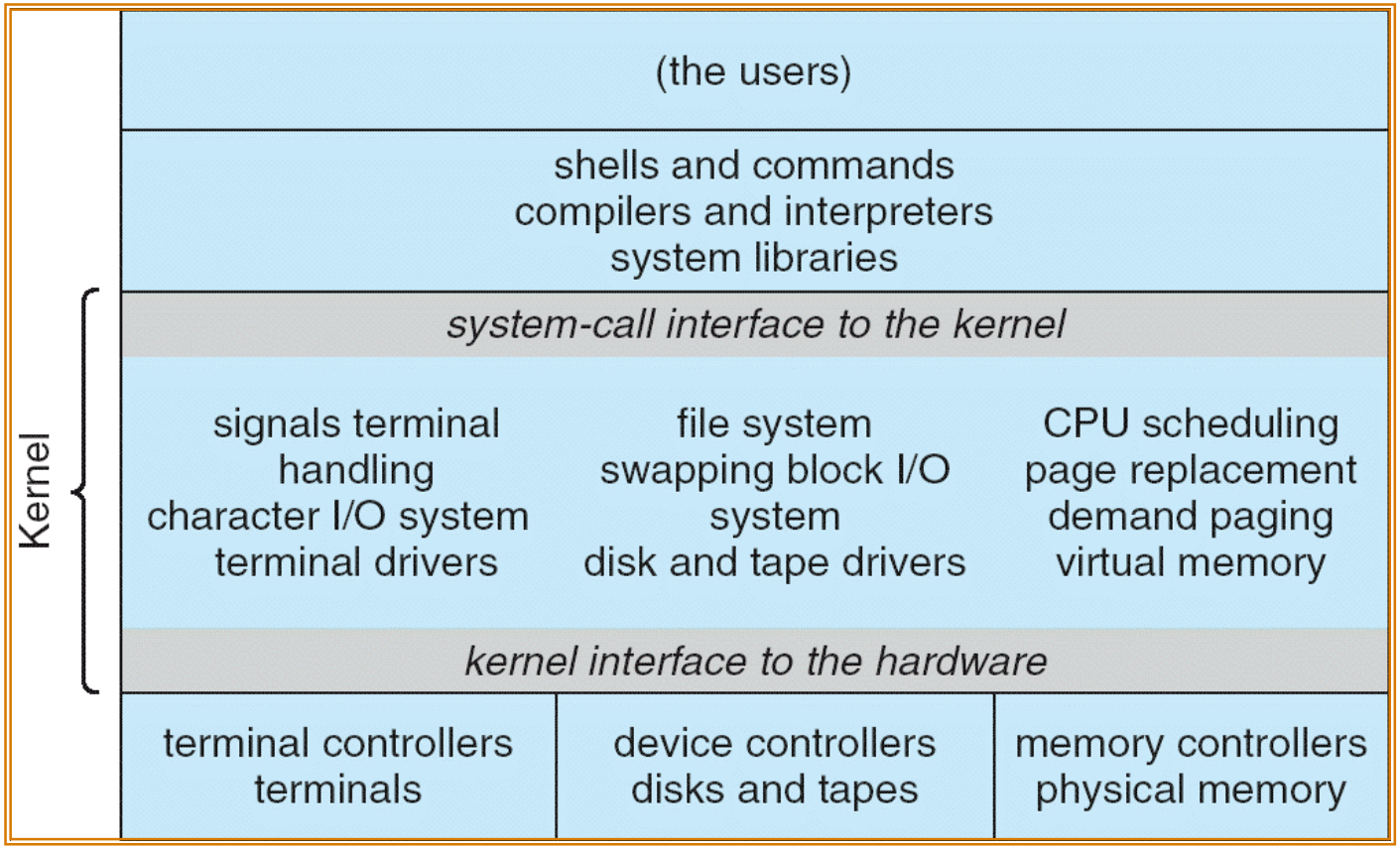
- MS-DOS – written to provide the most functionality in the least space
  - Not divided into modules
  - Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated



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
- UNIX – the 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

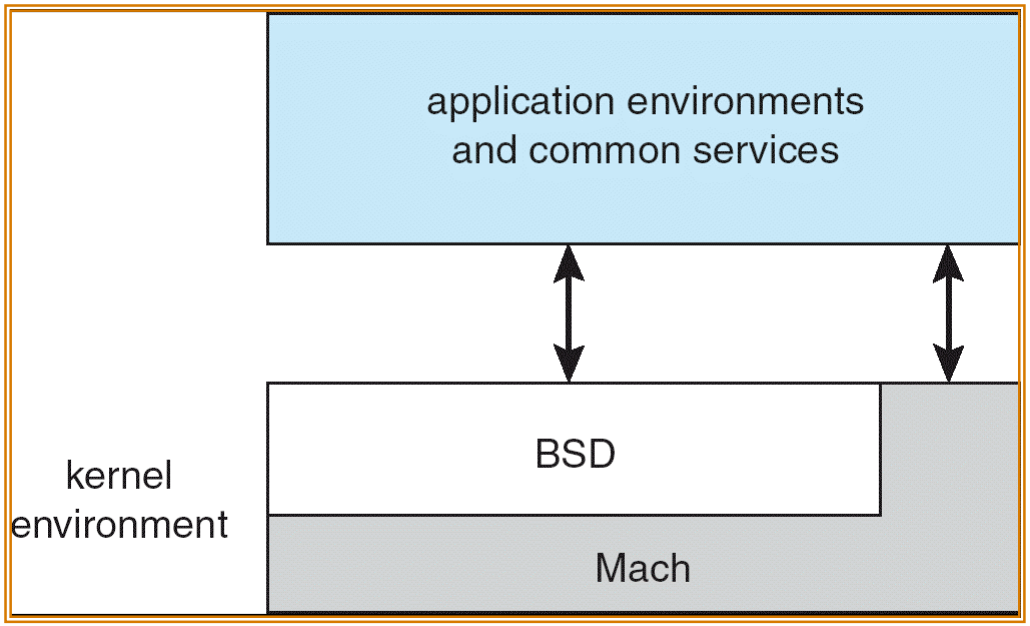
UNIX System Structure



Microkernel System Structure

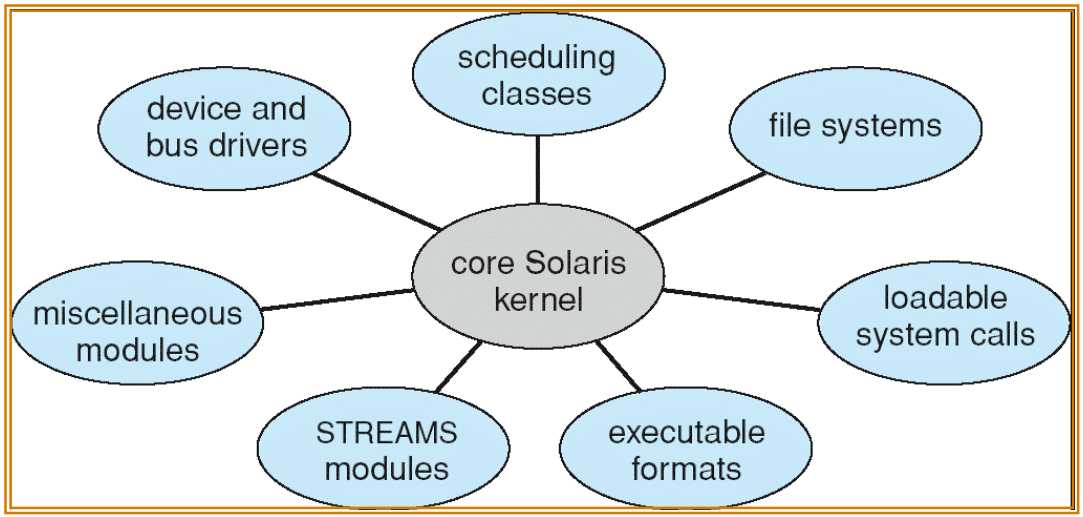
- Moves as much from the kernel into "*user*" space
- Communication takes place between user modules using message passing
- Benefits:
  - Easier to extend a microkernel
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure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

Mac OS X Structure



Modules

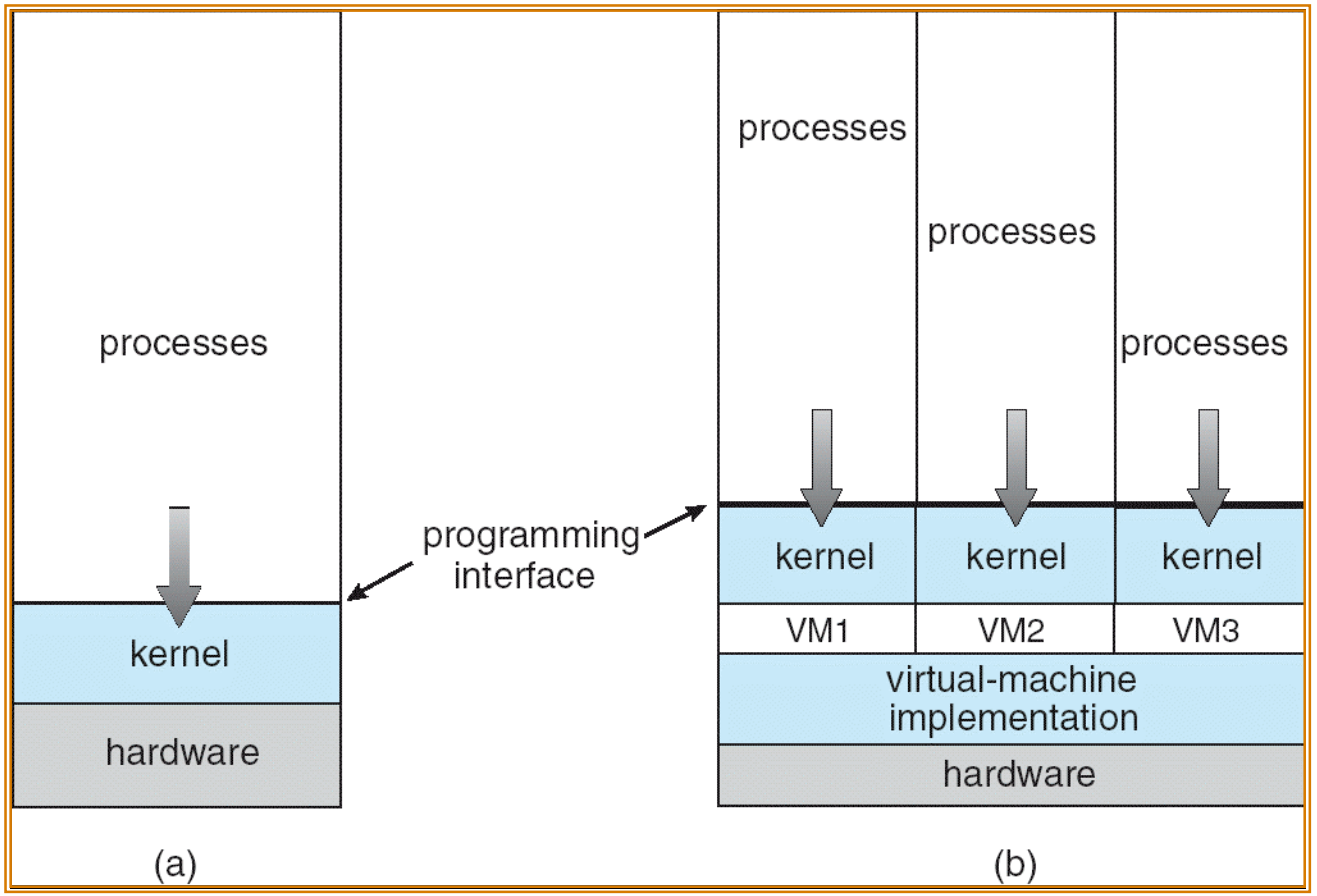
- Most modern operating systems implement kernel modules
  - Uses object-oriented approach
  - Each core component is separate
  - Each talks to the others over known interfaces
  - Each is loadable as needed within the kernel
- Overall, similar to layers but with more flexible



Virtual Machines

- A **virtual machine** takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware
- A virtual machine provides an interface identical to the underlying bare hardware
- The operating system creates the illusion of multiple processes, each executing on its own processor with its own (virtual) memory

Virtual Machines (Cont.)



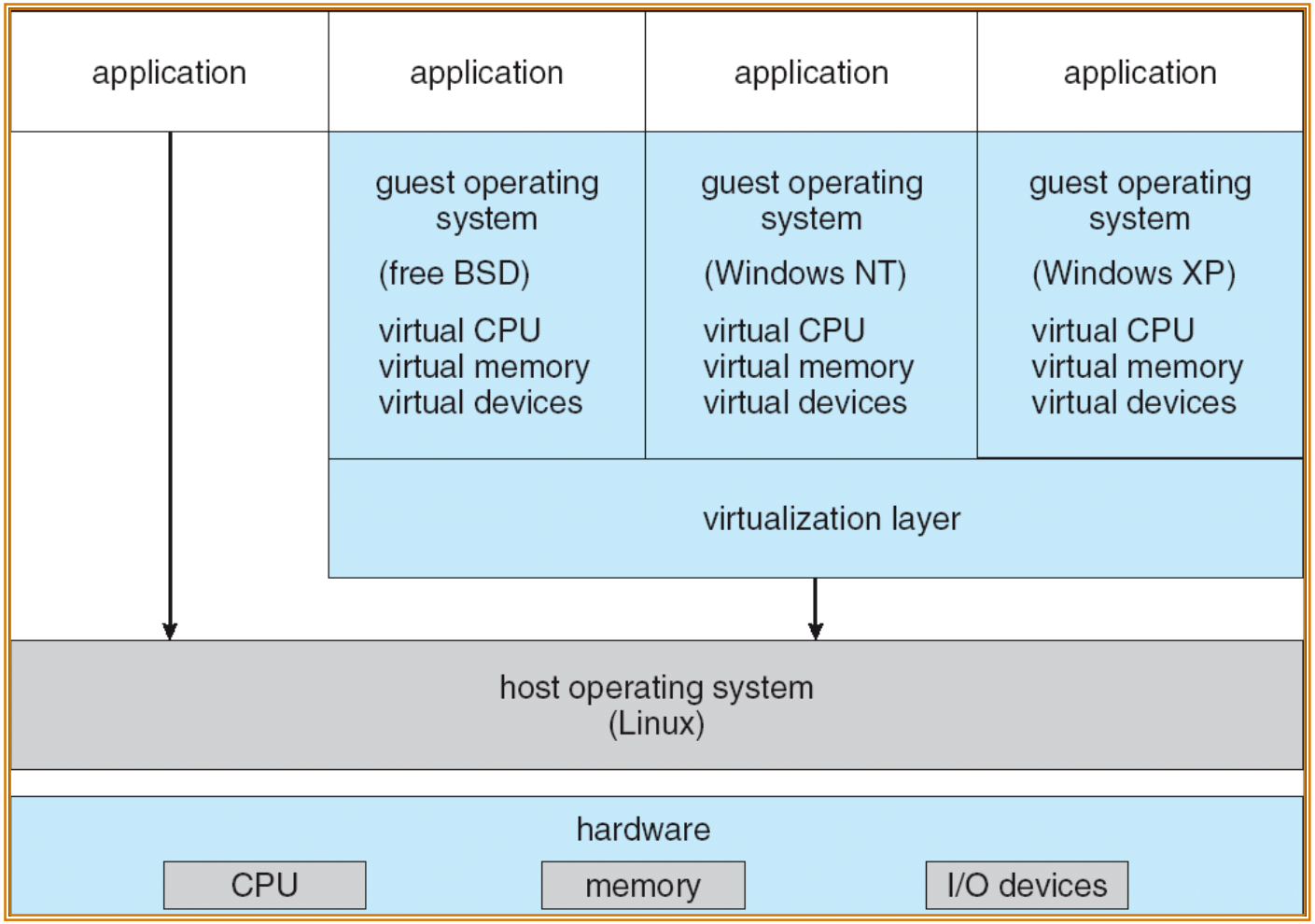
(a) Nonvirtual machine (b) virtual machine

Virtual machines for data centers

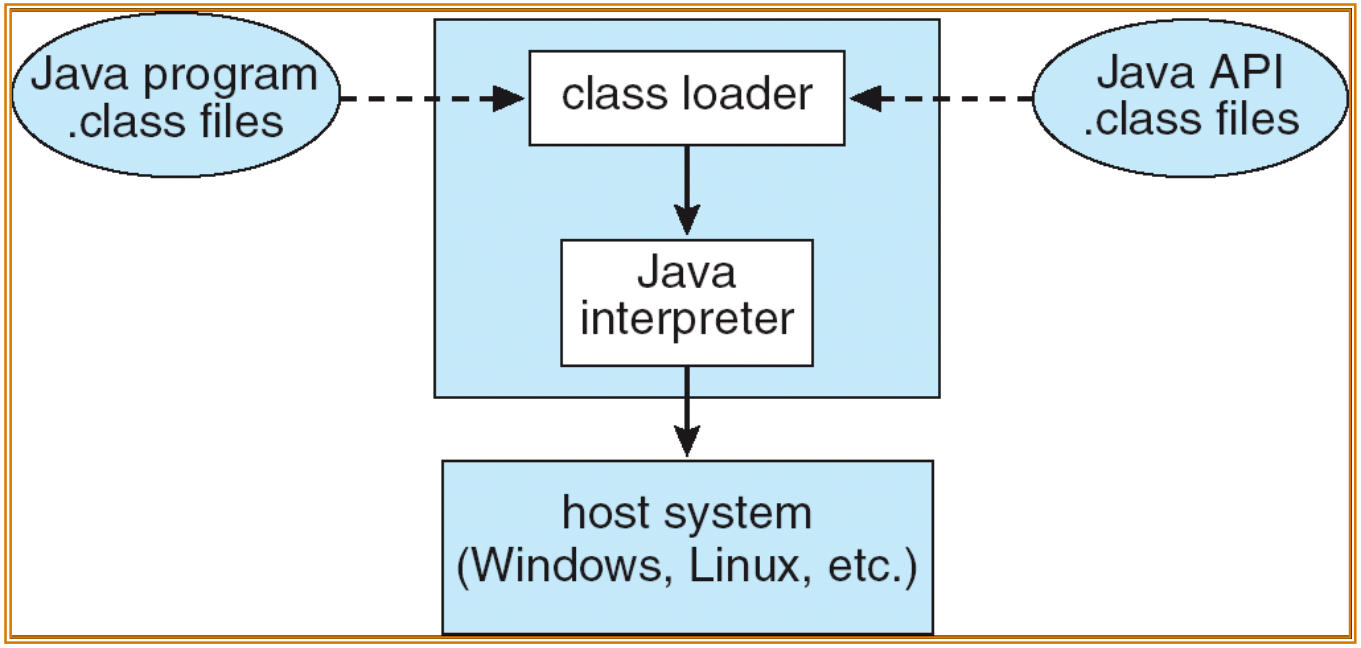
- The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
- A virtual-machine system is a perfect vehicle for operating-systems research and development.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

Using VM by IT - VMWare ACT

VMware Architecture



The Java Virtual Machine



Operating System Generation

- Operating systems are designed to run on any of a class of machines; the system must be configured for each specific computer site
- SYSGEN program obtains information concerning the specific configuration of the hardware system
- *Booting* – starting a computer by loading the kernel
- *Bootstrap program* – code stored in ROM that is able to locate the kernel, load it into memory, and start its execution

System Boot

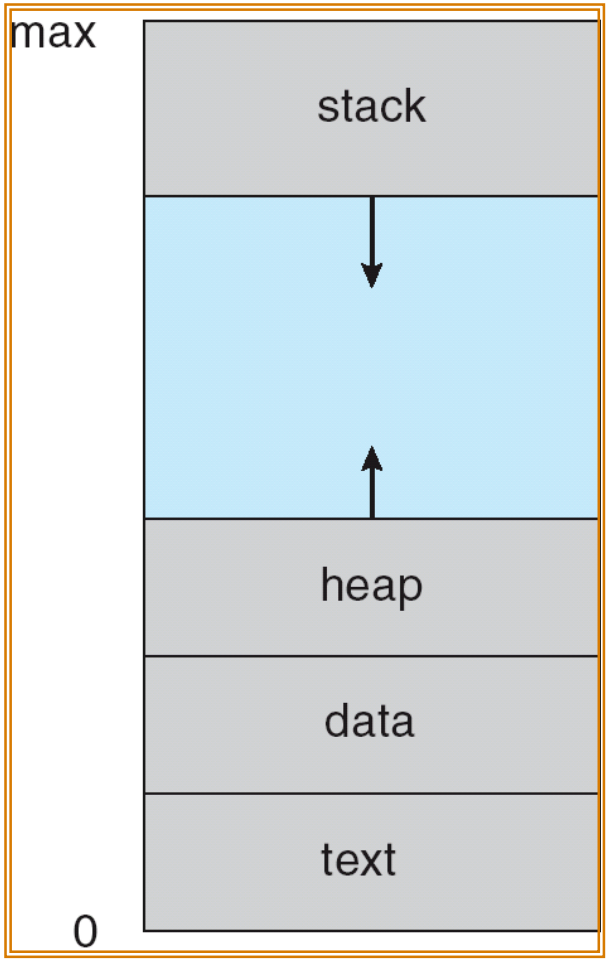
- Operating system must be made available to hardware so hardware can start it
  - Small piece of code – **bootstrap loader**, locates the kernel, loads it into memory, and starts it
  - Sometimes two-step process where **boot block** at fixed location loads bootstrap loader
  - When power initialized on system, execution starts at a fixed memory location
    - Firmware used to hold initial boot code

Wrapup

- System calls provide a mechanism for user programs to access OS services
  - System programs use system calls to provide functionality to users
- Need to design the OS for maximum flexibility of the user and OS developer
- Other issues such as bootstrapping to initialize the OS

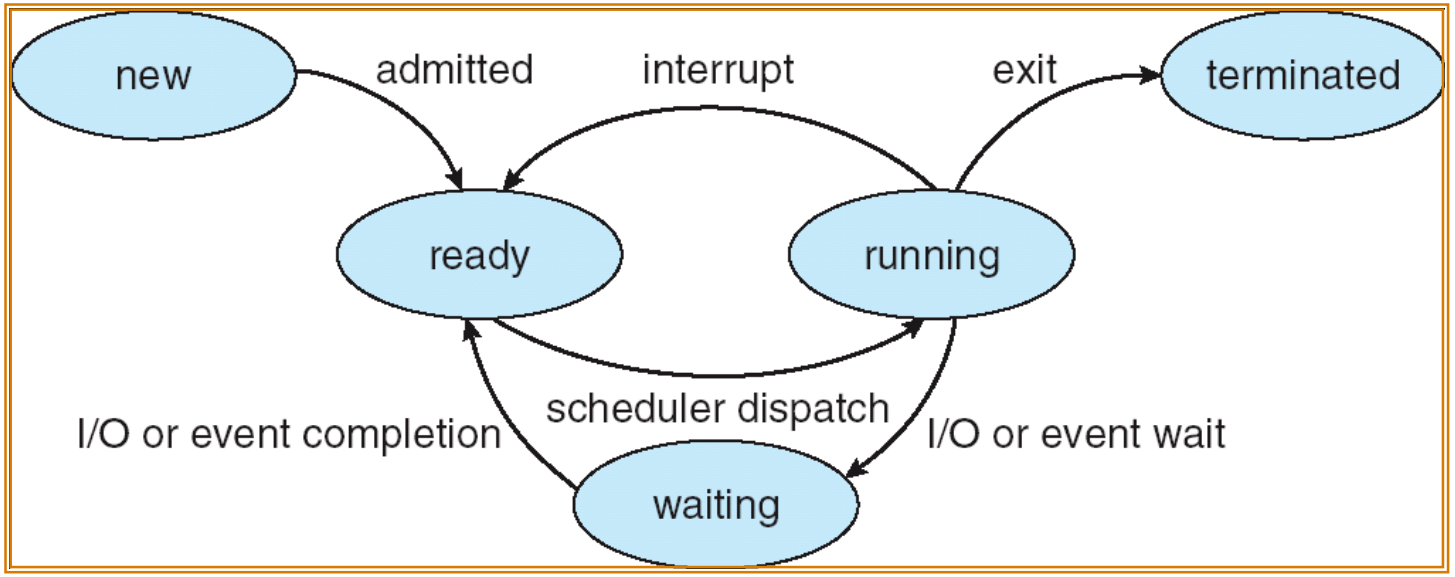
Chapter 3: Process Concept

- Process – a program (like MS Word) in execution; process execution must progress in sequential fashion
- A process includes:
  - program counter, register
  - Stack (temporary values, function parameters) , heap (memory allocations)
  - data section (global valuables), text section (code)



Process State

- As a process executes, it changes state
  - new: The process is being created
  - running: Instructions are being executed
  - waiting: The process is waiting for some event to occur
  - ready: process is waiting to be assigned to a processor
  - terminated: The process has finished execution



Process Control Block (PCB)

Information associated with each process and maintained by the operating system

- Process state
- Program counter
- CPU registers
- CPU scheduling information
- Memory-management information
- Accounting information
- I/O status information