操作系统原理及应用

李　伛
xchlw@seu.edu.cn

计算机科学与工程学院、软件学院
江苏省网络与信息安全重点实验室

Chapter 2 Operating-System Structures

Outline

- Operating System Functions
- Operating System Services
- Operating System Interfaces
- Operating System Structure
- Operating System Design and Implementation

Common Functions of OS

- Process Management
- Main Memory Management
- File Management
- Secondary-Storage Management
- I/O System Management

Process Management

- A *process* is a program in execution
- System Processes and User Processes
- Activities for process management
  - Process creation and deletion
  - process suspension and resumption
  - Provision of mechanisms for
    - process synchronization
    - process communication
    - Deadlock handling

Main-Memory Management

- Memory is a large array of words or bytes, each with its own address.
- It is generally the only large storage device that the CPU is able to address and access directly.
- Main memory is a volatile storage device.
- Activities for main-memory management
  - Record the usage of Main-memory
  - Allocate and deallocate memory space as needed

File Management (1/2)

- There are different types of physical media to store information. Each of them has its own characteristics and physical organization.
  - Access speed
  - Data-transfer rate
  - Access method (sequential or random)
- Operating System provides a uniform logical view of information storage, i.e., file.

File Management (2/2)

- A file is a collection of related information (programs and data) defined by its creator.
- Activities for file management
  - File creation and deletion
  - Directory creation and deletion
  - Support of primitives for manipulating files and directories
  - Mapping files onto secondary storage
  - File backup on stable (nonvolatile) storage media

Secondary-Storage Management

- Since main memory (*primary storage*) is volatile and too small to accommodate all data and programs permanently, the computer system must provide ***secondary storage* to back up main memory**.
- Most modern computer systems use **disks** as the principle on-line storage medium, for both programs and data.

Secondary-Storage Management

- Activities for disk management
  - Free space management
  - Storage allocation
  - Disk scheduling

I/O System Management

- Hiding the peculiarities of specific hardware devices from the user
- The I/O subsystem consists of
  - A memory-management component including buffer, caching, spooling
  - A general device-driver interface
  - Drivers for specific hardware devices

Outline

- Operating System Functions
- Operating System Services
- Operating System Interfaces
- Operating System Structure
- Operating System Design and Implementation

Operating System Services(1/2)

- Services for helping Users
  - Program execution – system capability to load a program into memory and to run it.
  - I/O operations – since user programs cannot execute I/O operations directly, the operating system must provide some means to perform I/O.
  - File-system manipulation – program capability to read, write, create, and delete files.
  - Communications – exchange of information between processes (shared memory or message passing)
  - Error detection – ensure correct computing by detecting errors in hardwares or in user programs.

Operating System Services(2/2)

- **Services for ensuring system operations**
  - **Resource allocation – allocating resources to multiple users or multiple jobs running at the same time.**
  - **Accounting – keep track of which users use how much and what kinds of computer resources.**
  - **Protection – ensuring that all access to system resources is controlled and recording all the connections for detection of break-ins.**

Outline

- Operating System Functions
- Operating System Services
- Operating System Interfaces
- Operating System Structure
- Operating System Design and Implementation

Operating System Interfaces(1/3)

- **Interfaces to Users**
  - **Command-Line Interface – text commands**
  - **Batch Interface – files including some commands**
  - **Graphical User Interface – window system**
    - **Mouse-based window and menu**
  - **Many systems now include both CLI and GUI interfaces**
    - **Microsoft Windows is GUI with CLI "command" shell**
    - **Apple Mac OS X as "Aqua" GUI interface with UNIX kernel underneath and shells available**
    - **Linux is CLI with optional GUI interfaces (Java Desktop, KDE)**

Operating System Interfaces(2/3)

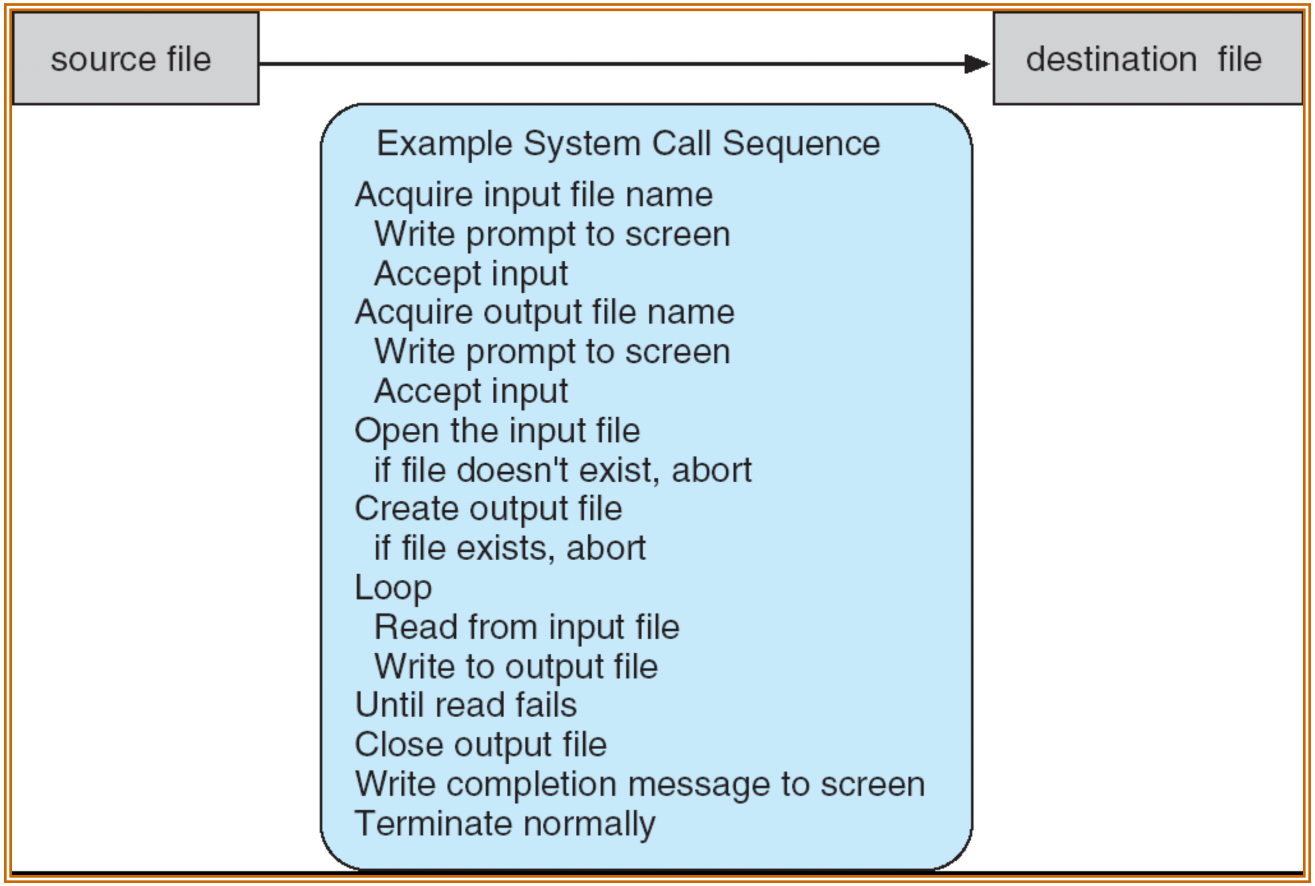
- **Command-line Interpreter (Shell)**
  - **The program that reads and interprets control statements**
  - **DOS OS: command.com**
- **Two ways to implement commands**
  - **Internal Command: The command interpreter itself contains the code to execute the command. (dir, copy)**
  - **External Command: System programes implement most commands. (fdisk, format)**

Operating System Interfaces(3/3)

- Interfaces to Programes
  - System calls
    - Typically written in assembly-language instructions or a high-level language (C or C++)
    - Mostly accessed by programs via a high-level Application Program Interface (API) rather than direct system call use

Example of System Calls

- **System call sequence to copy the contents of one file to another file**



System Call Implementation

- **System-call Interface (SCI)**
  - **Typically, a number associated with each system call**
  - **System-call interface maintains a table indexed according to these numbers**
  - **The system call interface invokes intended system call in OS kernel and returns status of the system call and any return values**

System Call Implementation

- **Application Programming Interface (API)**
  - **API functions invoke the actual system calls on behalf of the application programmer**
  - **Just needs to obey API and understand what OS will do as a result call, most details of OS interface hidden from programmer by API**
  - **APIs are Managed by run-time support library (set of functions built into libraries included with compiler)**

System Call Implementation

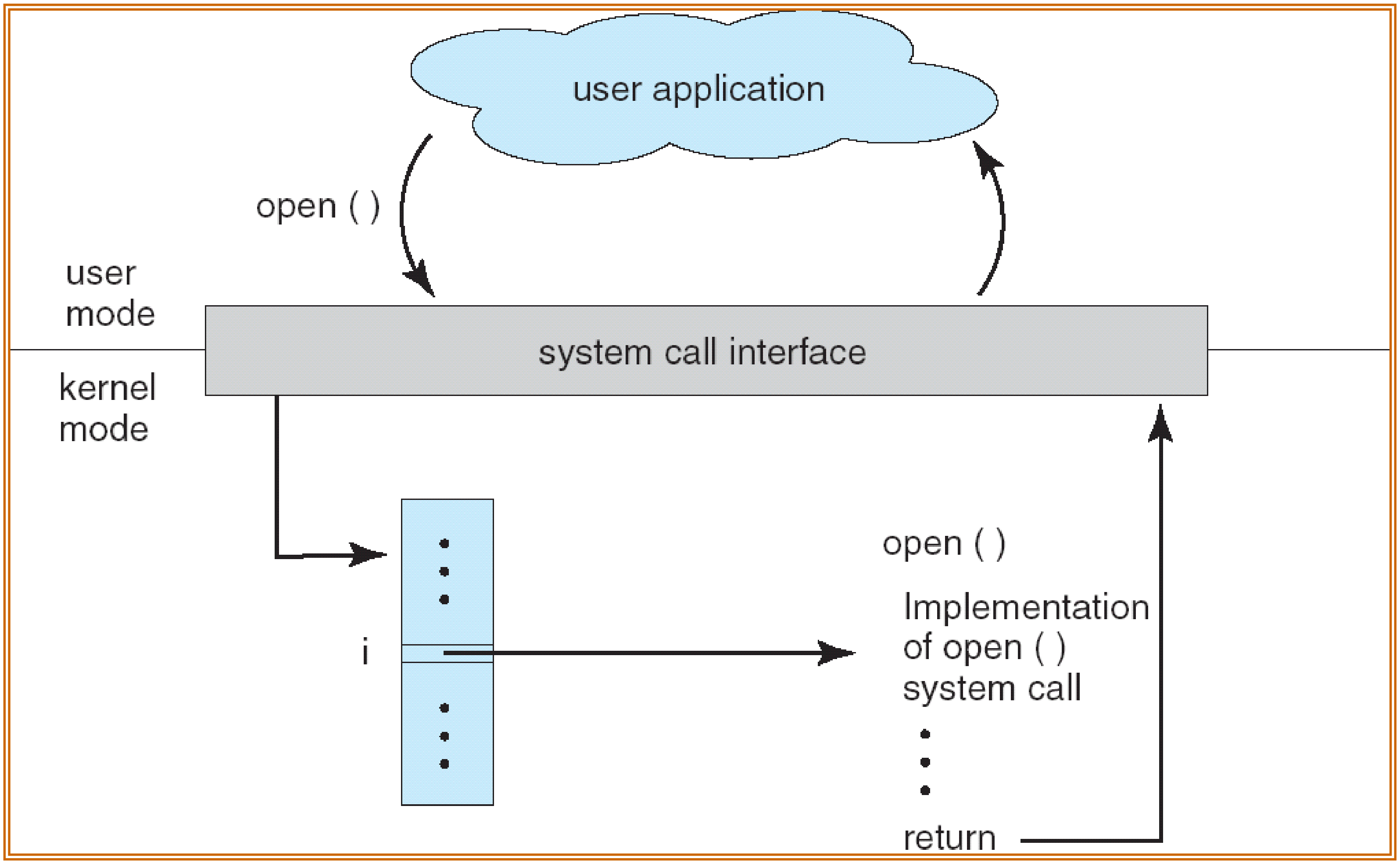
- *Discuss*

Why do user use APIs rather than system calls?

System Call Implementation

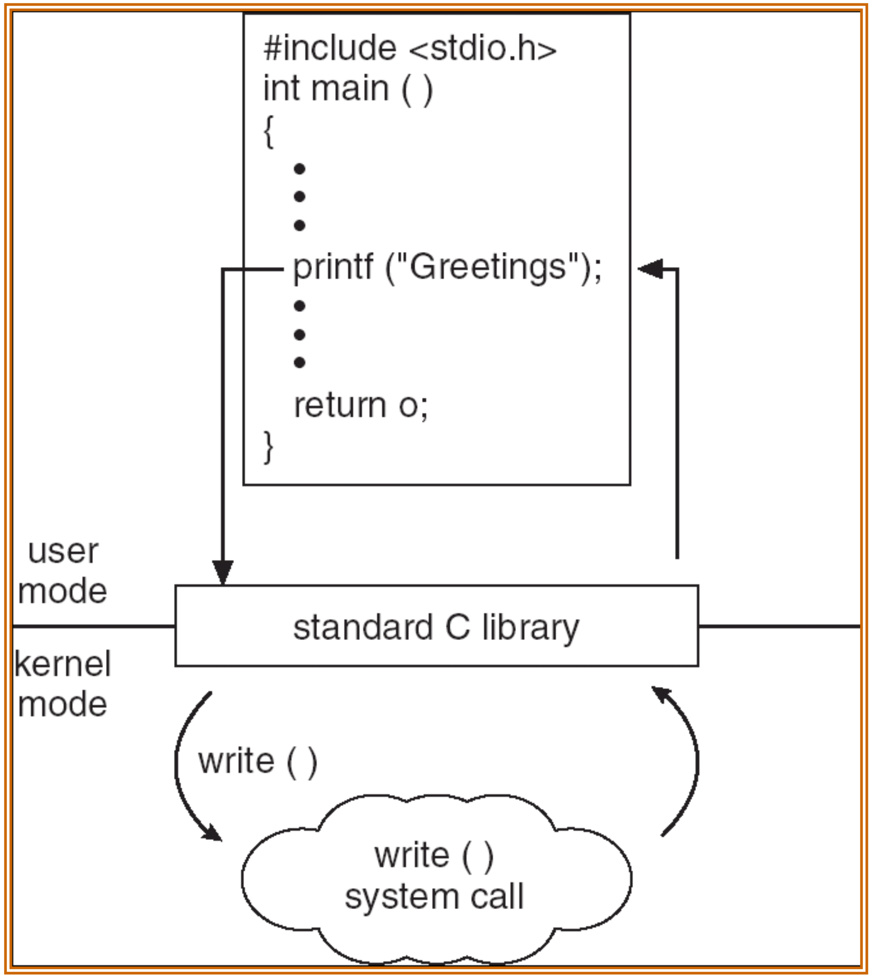
- Three most common APIs
  - Win32 API for Windows
  - POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X)
  - Java API for the Java virtual machine (JVM)

Relationship between API, SCI, OS



Standard C Library Example

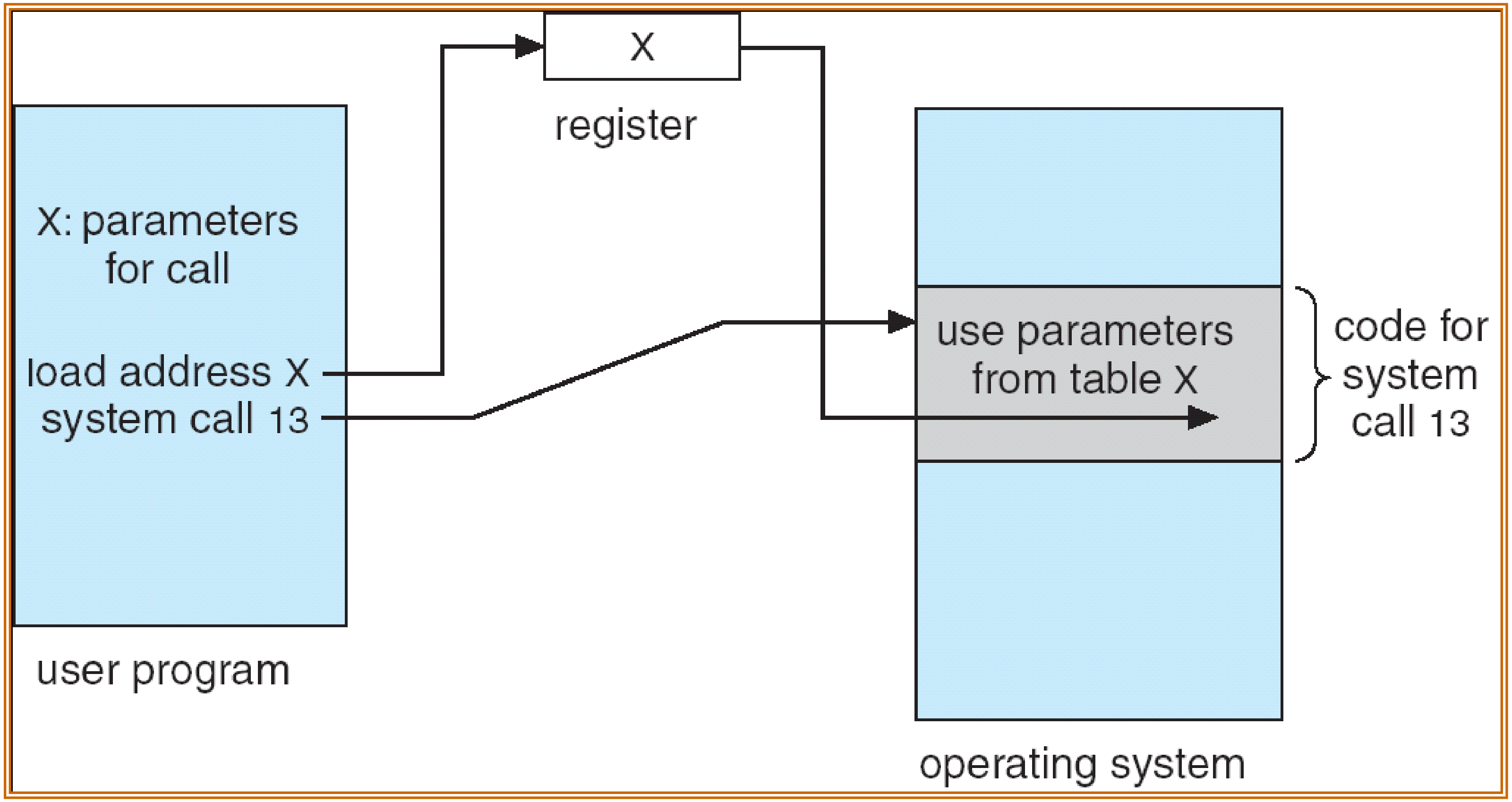
- **C program invoking printf() library call, which calls write() system call**



System Call Parameter Passing

- Often, more information is required than simply identity of desired system call
- Three general methods used to pass parameters to the OS
  - Simplest: pass the parameters in ***registers***
  - Parameters stored in a ***block,*** **or table, in memory**, and **address** of block passed as a parameter **in a register**
  - Parameters placed, or ***pushed,*** **onto the *stack* by the program** and ***popped*** **off the stack by the operating system**

Parameter Passing via Table



Types of System Calls

- Process control
- File management
- Device management
- Information maintenance
- Communications

Types of System Calls

- **Process control**
  - **end, abort**
  - **load, execute**
  - **create and terminate process**
  - **get and set process attributes**
  - **wait for time**
  - **wait event, signal event**
  - **allocate and free memory**

Types of System Calls

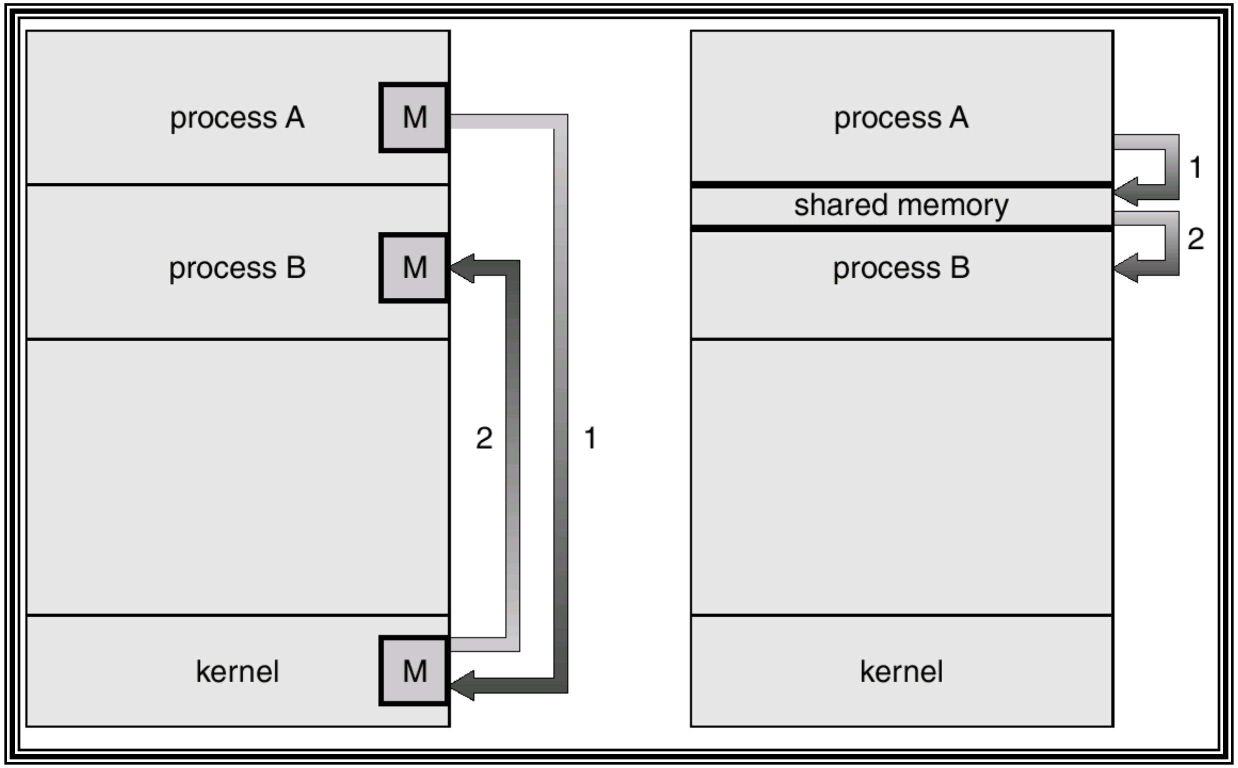
- File management
  - create and delete file
  - open and close file
  - read, write, reposition
  - get and set file attributes
- Device management
  - request and release file
  - read, write, reposition
  - get and set device attributes
  - logically attach or detach devices

Types of System Calls

- Information maintenance
  - get and set time or date
  - get and set system data
  - get and set process, file or device attributes
- Communications
  - create and delete communication connection
  - send and receive message
  - transfer status information
  - attach or detach remote devices

Communication Models

- Communication may take place using either **message passing** or **shared memory**.



Outline

- Operating System Functions
- Operating System Services
- Operating System Interfaces
- Operating System Structure
- Operating System Design and Implementation

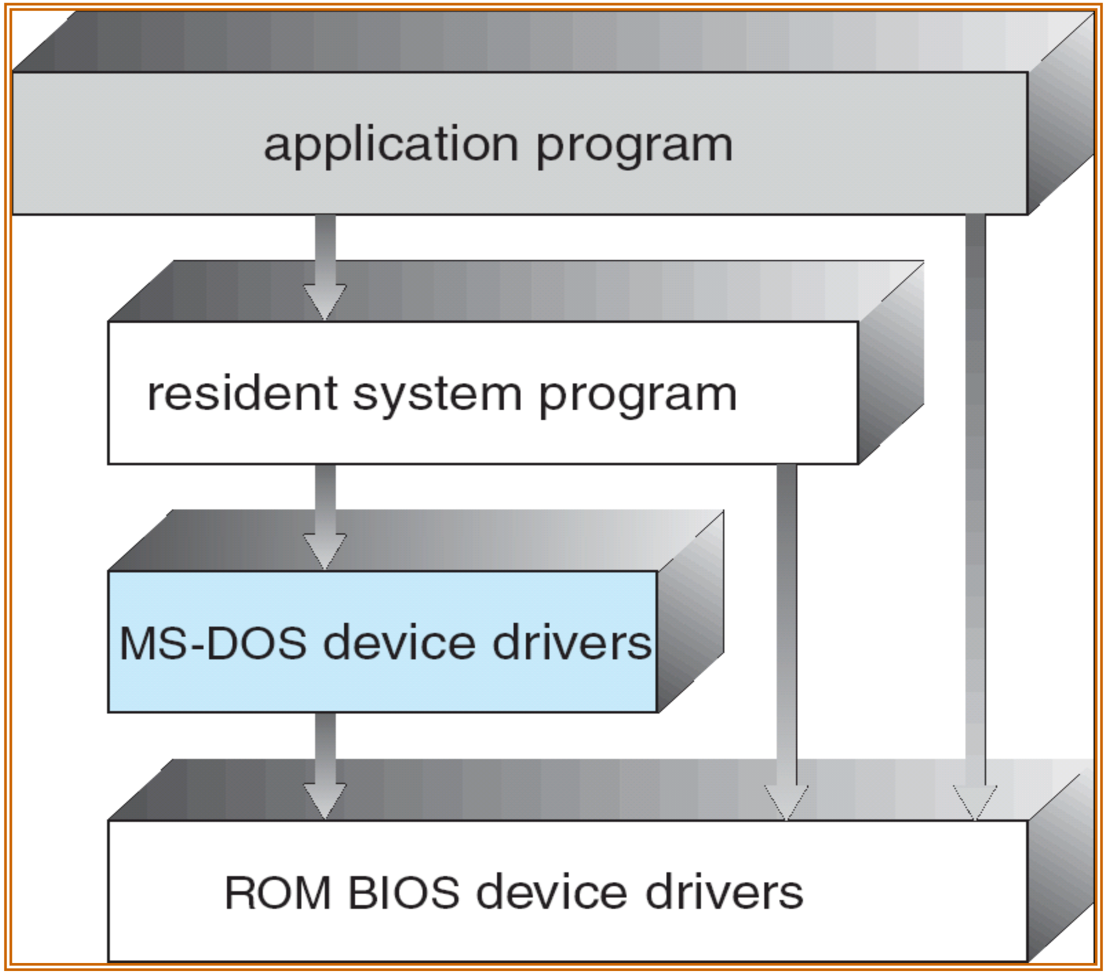
Operating System Structure

- Simple Structure
- Layered Structure
  - Virtual Machines
- Microkernel Structure
- Modules

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

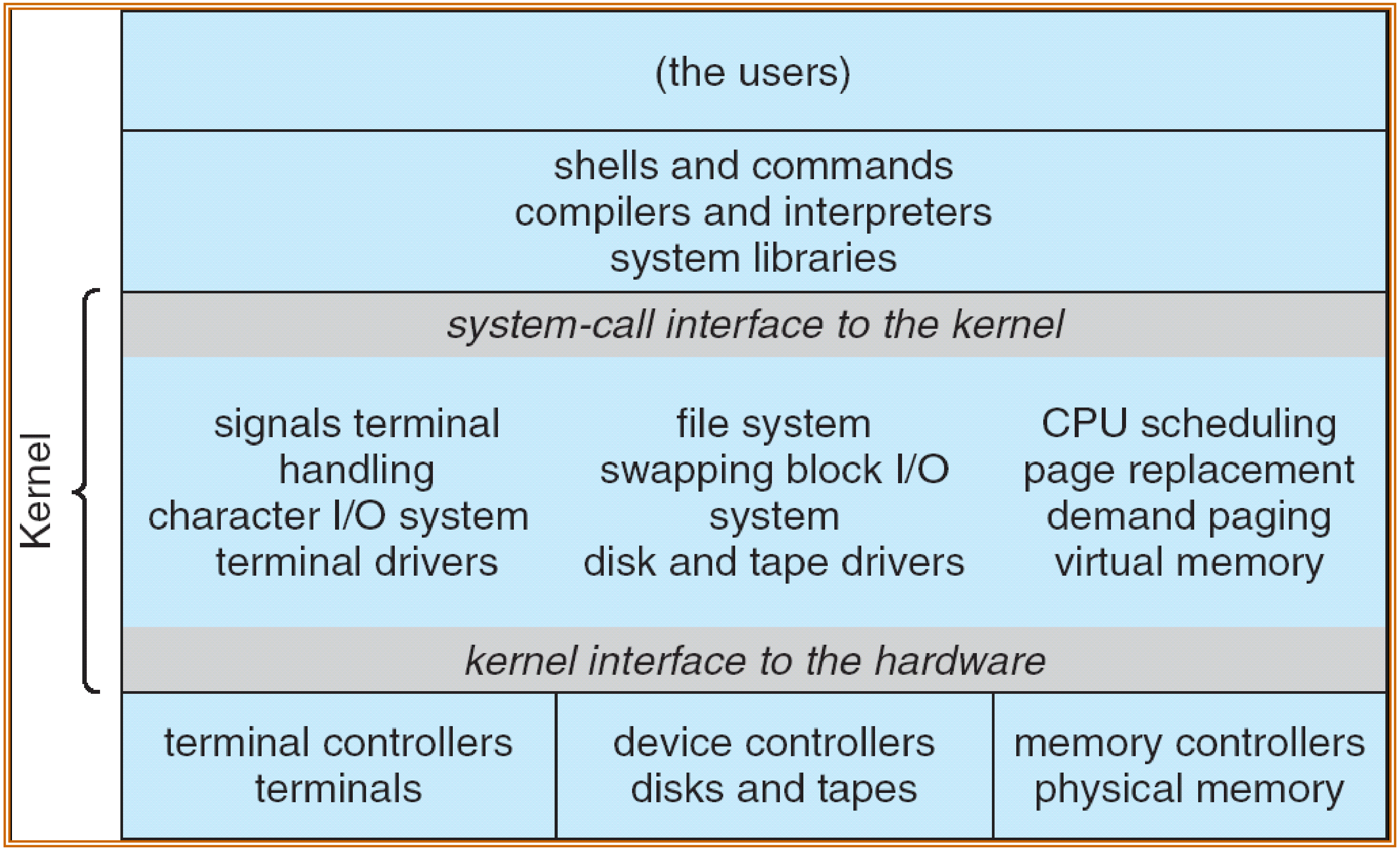
MS-DOS Layer Structure



Simple Structure

- Original UNIX
  - Two separable parts
    - Systems programs
    - The kernel: Consists of everything below the system-call interface and above the physical hardware, provides a large number of functions for one level

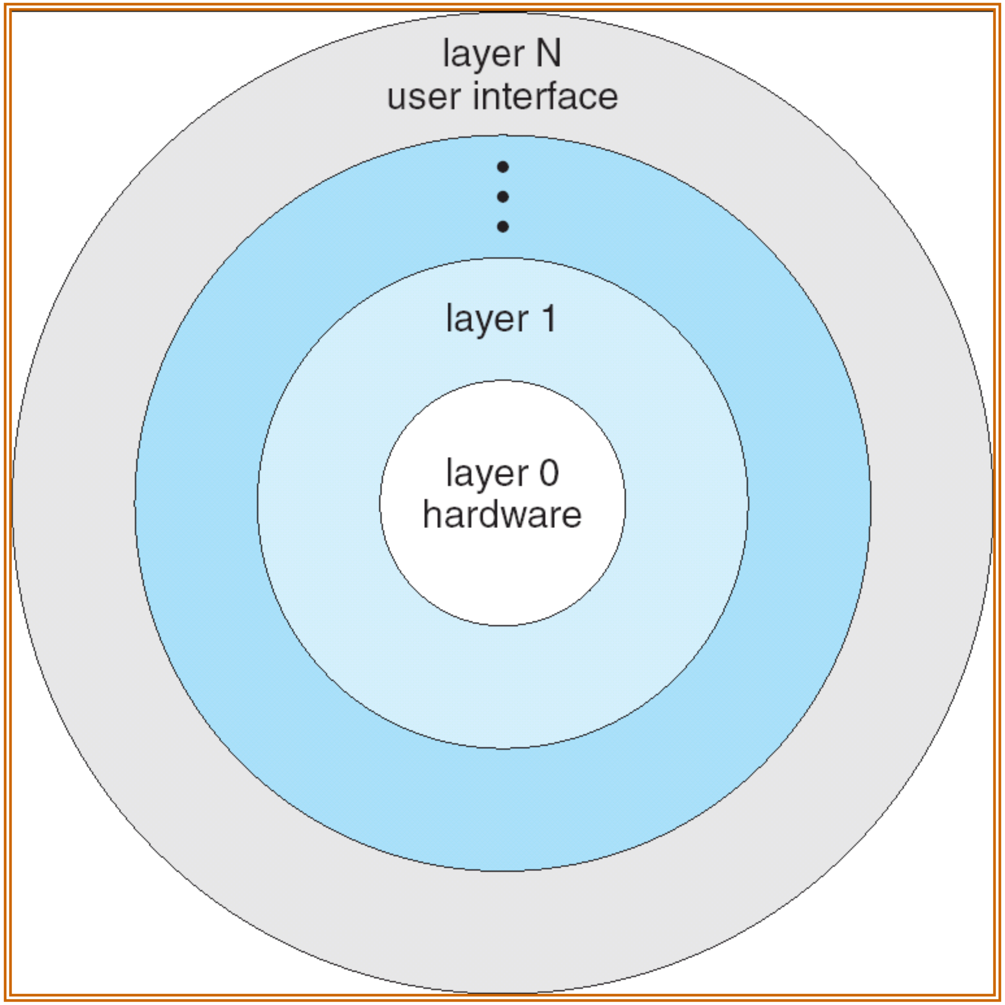
UNIX System Structure



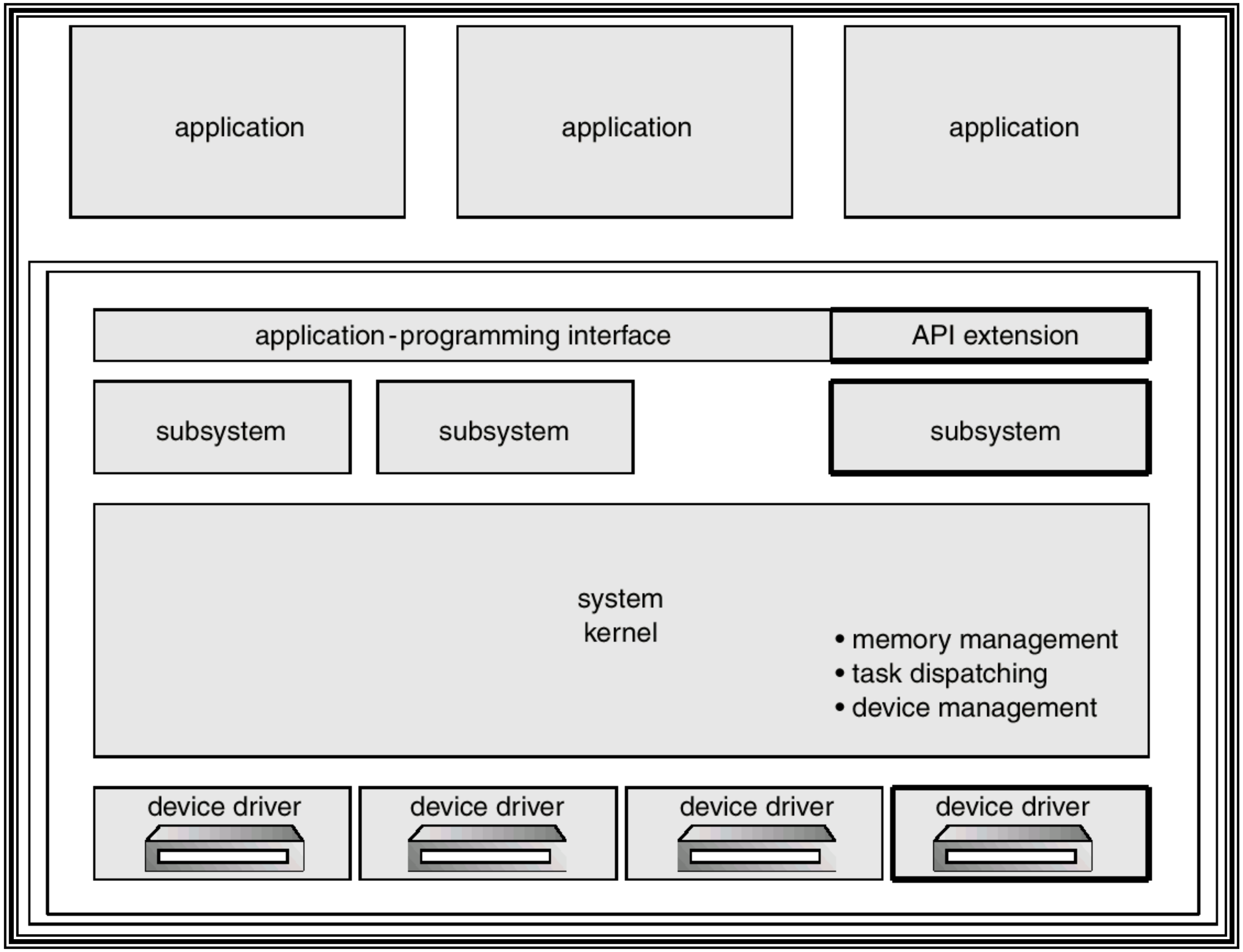
Layered Structure

- 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



OS/2 Layer Structure



Microkernel Structure

- Moves as much from the kernel into "*user*" space
- Typically, microkernels provide minimal process management, memory management and communication facility
- Communication takes place between user modules using message passing

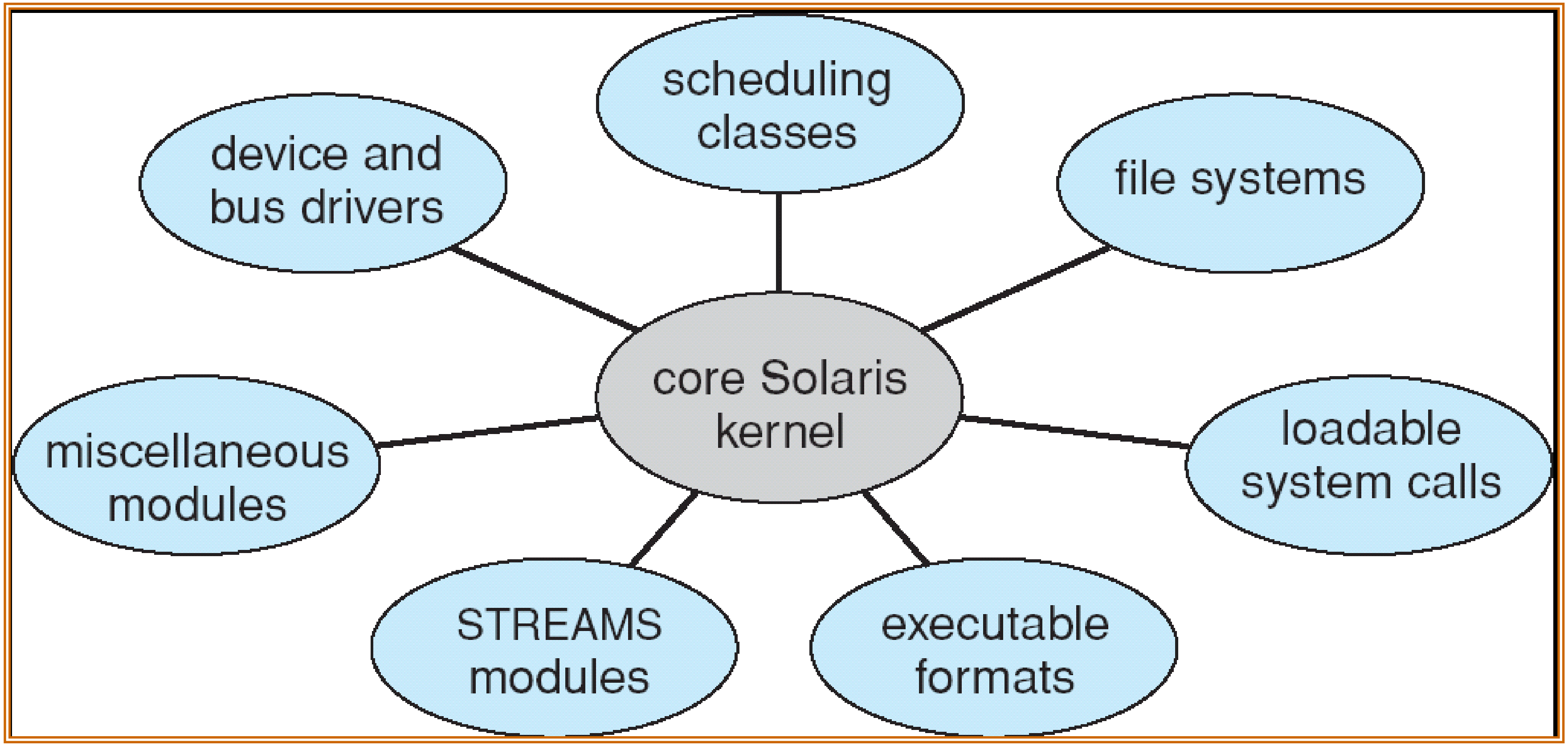
Microkernel Structure

- Benefits
  - Easier to extend a operating system
  - Easier to port the operating system to new architectures
  - More reliable (less code is running in kernel mode)
  - More secure
- Lacks
  - Performance overhead of user space to kernel space communication

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

Solaris Modular Approach



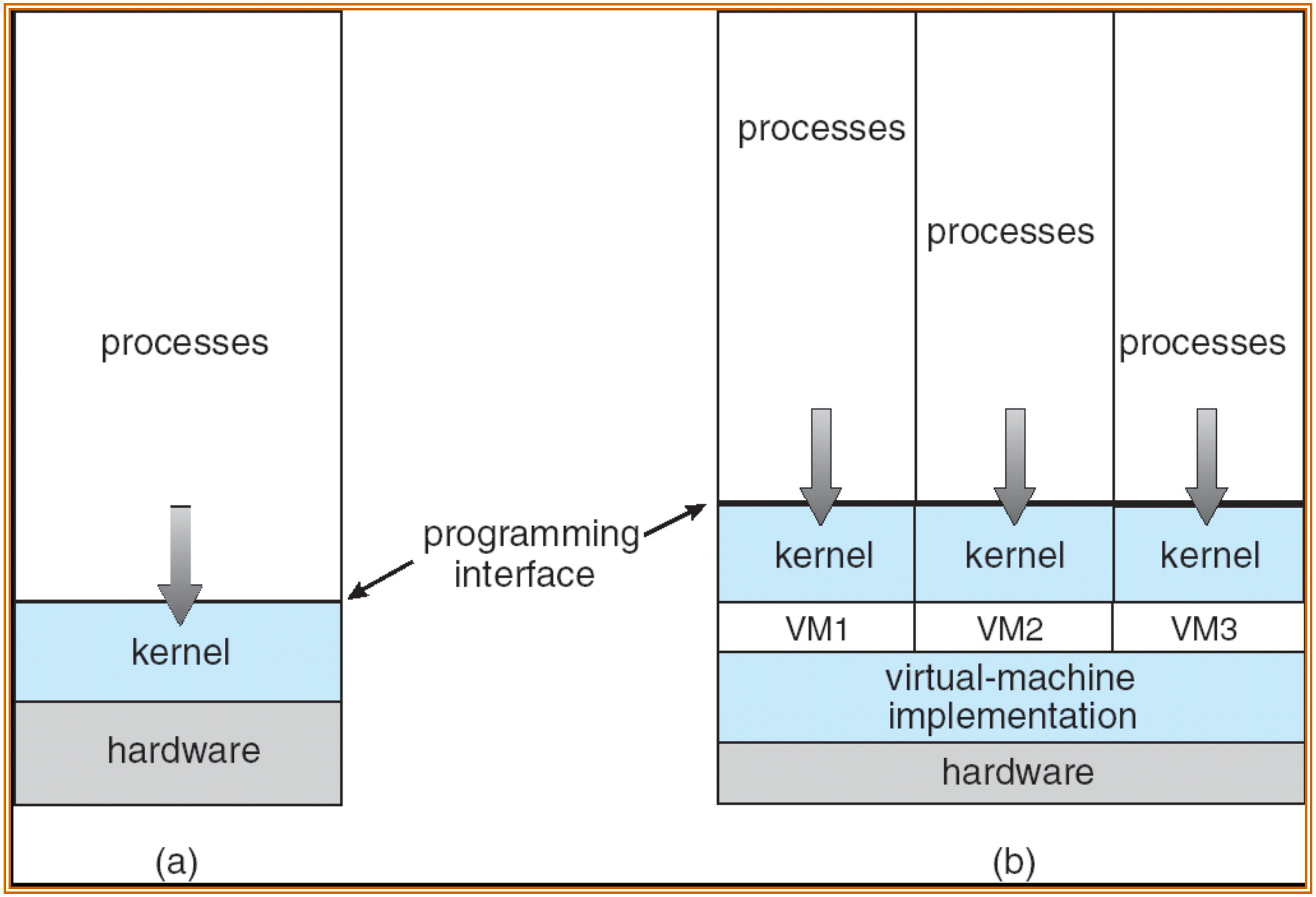
作业1

设计操作系统时采用的模块化内核方法和分层方法在那些方面类似？哪些方面不同？

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



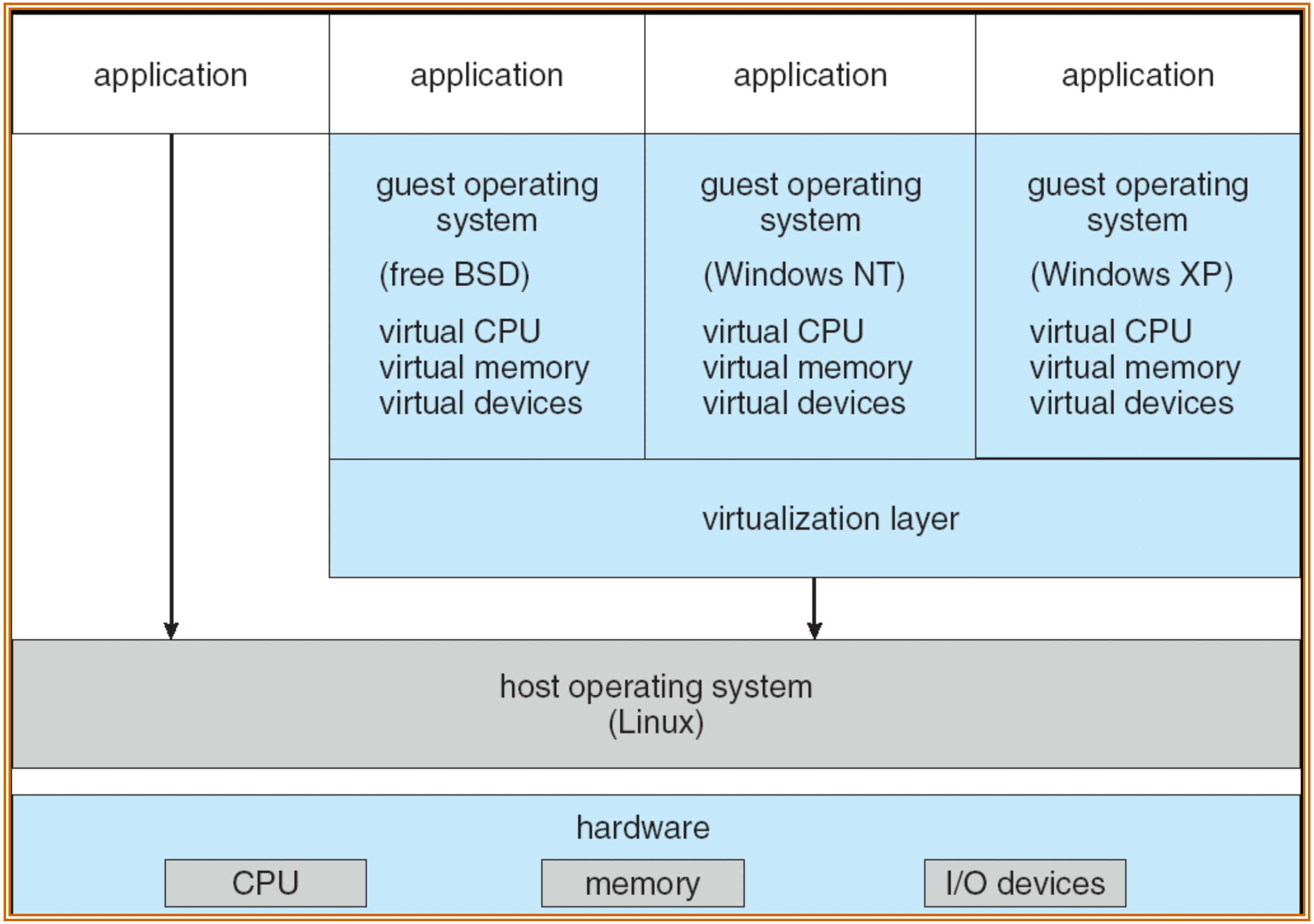
Virtual Machines

- 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

- Advantages
  - Providing complete protection of system resources
    - Each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.
  - Being able to share the same hardware yet run different operating systems concurrently
  - A perfect vehicle for operating-systems research and development
    - System development is done on the virtual machine, so does not disrupt normal system operation.

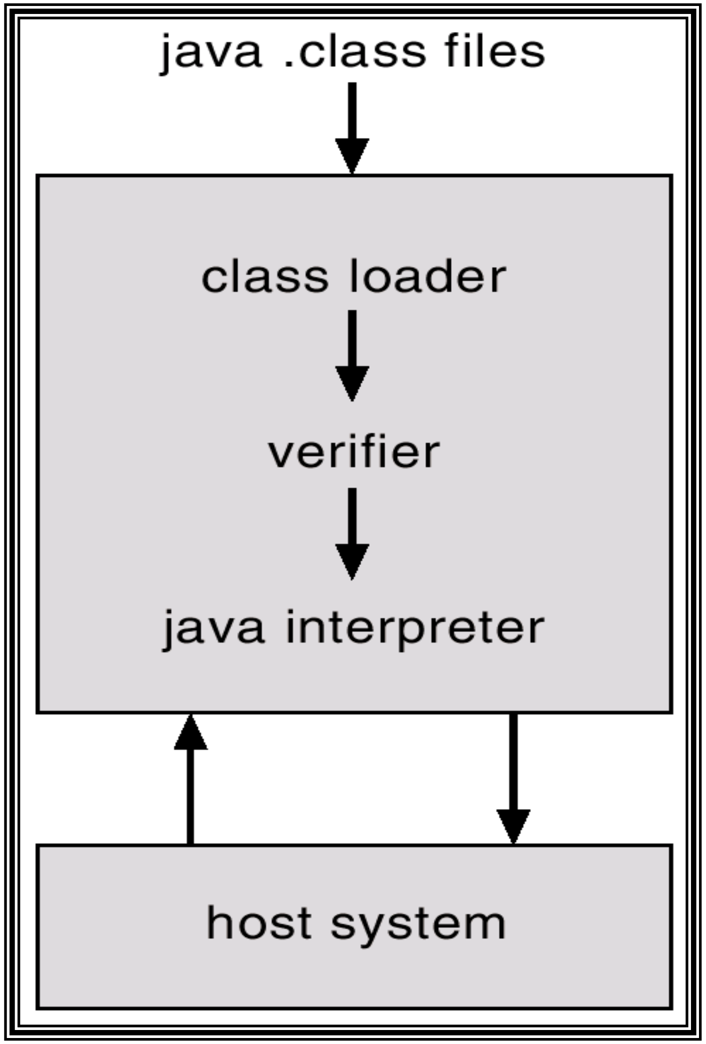
VMware Architecture



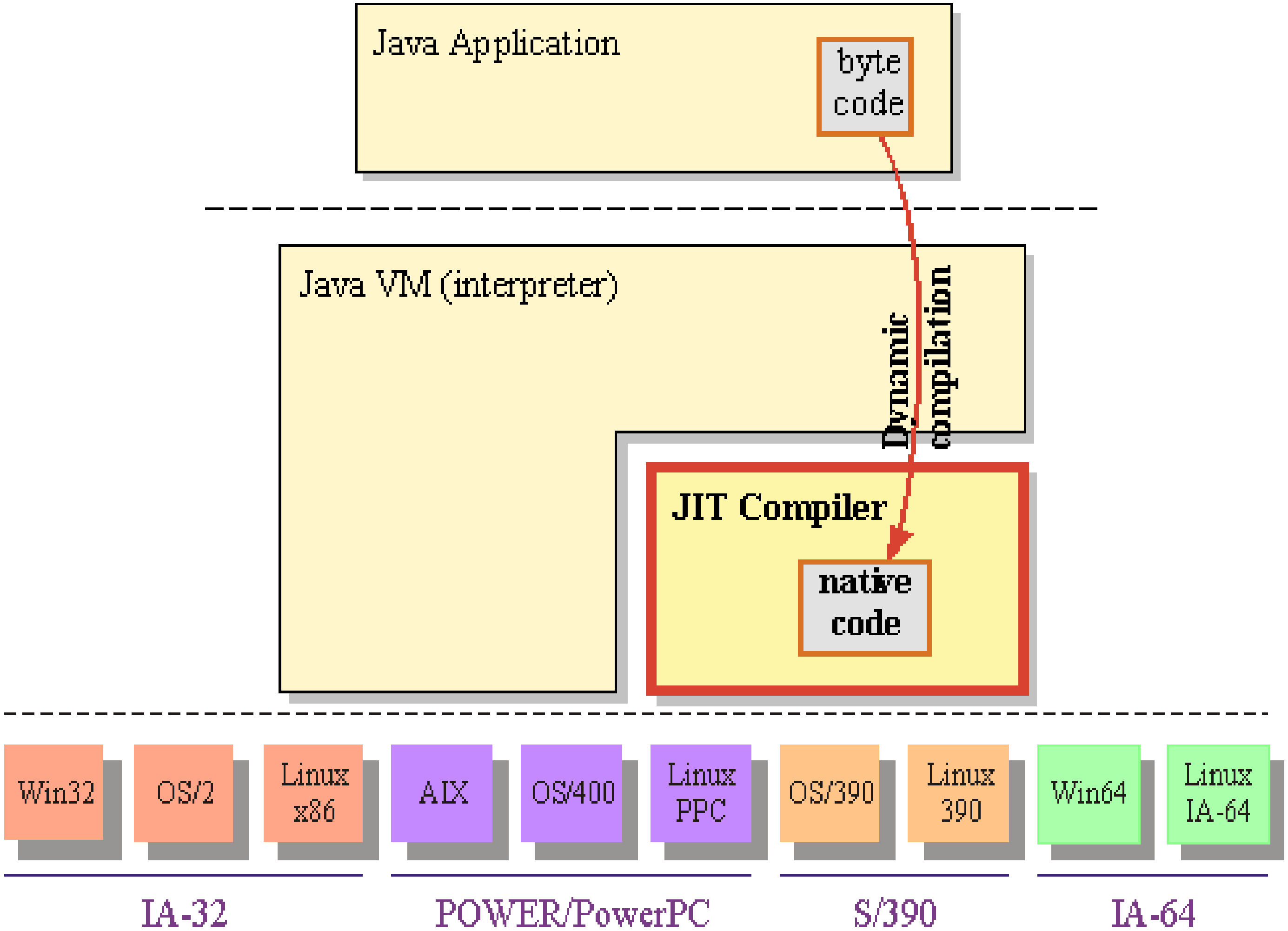
Java Virtual Machine

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

Java Virtual Machine



Java Virtual Machine



Outline

- Operating System Functions
- Operating System Services
- Operating System Interfaces
- Operating System Structure
- Operating System Design and Implementation

Operating System Design and Implementation

- Design and Implementation of OS not "solvable", but some approaches have proven successful
- Internal structure of different Operating Systems can vary widely
- Start by defining goals and specifications
- Affected by choice of hardware, type of system

Operating System Design and Implementation

- Design 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, as well as flexible, reliable, error-free, and efficient

Operating System Design and Implementation

- 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

Operating System Design and Implementation

- Written mostly in C or C++
  - Advantage — being far easier to port
  - Disadvantage — reduced speed and increased storage requirements
- Some small sections of assembly code for device drivers and for saving and restoring the state of registers

Part 1 小结（1/2）

- 操作系统概念（管理各种资源、支持程序运行、方便用户使用的程序集）
- 操作系统的基本目标（方便性与高效性）
- 引导程序、中断、中断处理程序、中断向量
- 存储结构：内存（小、易失）、二级存储（大、非易失）、分层结构
- I/O结构：设备控制器（本地缓冲）、DMA
- 硬件保护：双重模式操作、特权指令、I/O保护、内存保护、CPU保护

Part 1 小结（2/2）

- 操作系统的发展（大型机（无OS、批处理、多道程序设计（并发性、共享性、虚拟性、异步性）、分时）——桌面——并行（紧耦合）——分布式（松耦合，集群）——专用（实时、手持））
- 操作系统的功能：进程（CPU）管理、内存管理、文件管理、磁盘管理、I/O管理、用户接口
- 操作系统的服务：程序执行、I/O操作、文件系统操作、通信、错误检测与处理、资源分配、统计、保护
- 操作系统的接口：用户接口（CLI、GUI）+程序接口（系统调用（参数传递、类型）、SCI、API）
- 操作系统的结构：简单结构、分层结构（虚拟机）、微核结构（进程管理、内存管理、通信功能）、模块化