操作系统结构

俯视 OS

- 操作系统做什么的？
- 硬件系统的组成
- 计算机系统的体系结构
- 现代操作系统的特征
- 操作系统的服务类别
- 操作系统用户界面
- 系统调用
- 系统程序
- 操作系统结构

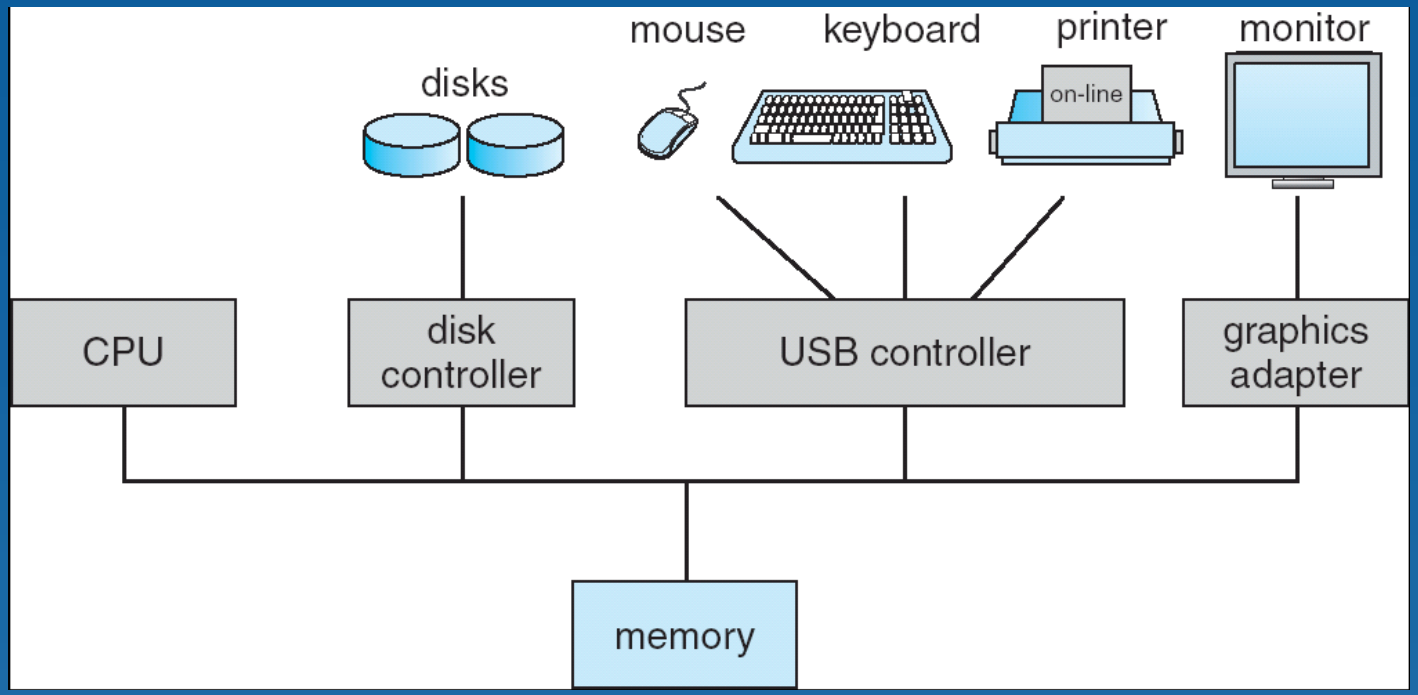
操作系统做什么的 ？

- 操作系统定义（如果一定要一个？）
- A program that acts as an intermediary between a user of a computer and the computer hardware

- 操作系统的目标
  - 使用户方便地使用计算机
- 使计算机硬件高效率运行

硬件系统的组成

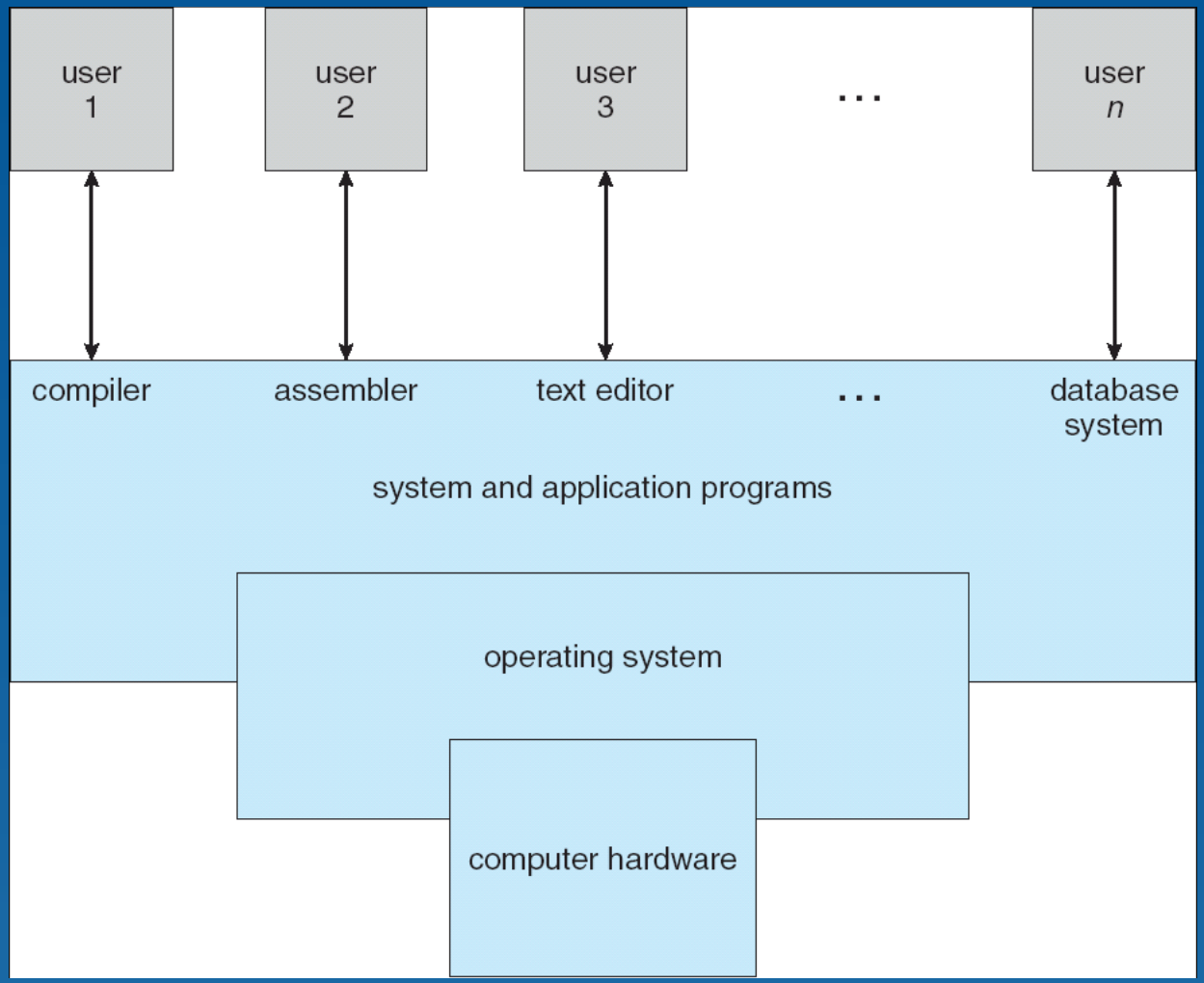
- 如图所示的硬件设施
  - 一个或数个 CPUs，加上一些设备控制器，通过内部总线连接在一起。它们共享内存
  - 这些 CPUs 和设备并行执行，并且竞争使用内存的访问周期



计算机系统的体系结构

- 计算机系统从下层到上层共有 4 层
  - 硬件 – 提供基本的计算资源
    - CPU，内存， I/O 设备等
  - 操作系统
  - 应用程序
  - 用户
    - 人，机器设备，网络上的其它计算机

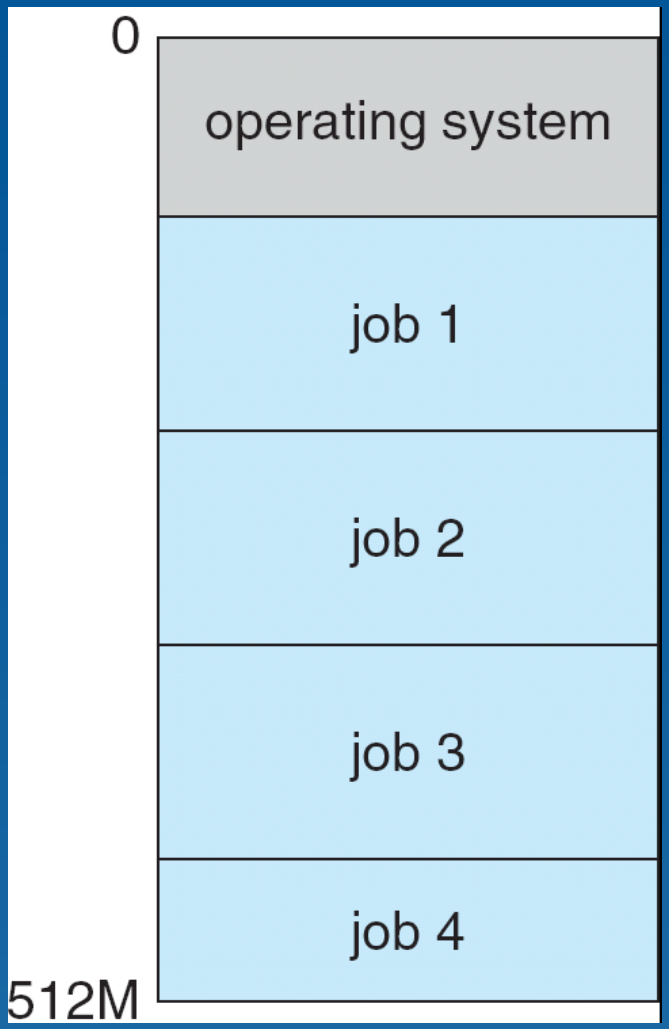
计算机系统的 4 个层次


user 1
user 2
user 3
...
user n
compiler
assembler
text editor
...
database system
system and application programs
operating system
computer hardware

现代操作系统的特征

- 多程序， Multiprogramming
  - 单用户系统有先天缺陷，无法让 CPU 和 I/O 设备同时处于运转状态
  - 多程序思想让多个程序竞争使用 CPU，使 CPU 总是有用户程序执行
  - 作业调度器每次选择一个作业，交 CPU 执行
  - 当这个作业被迫等待时（例如有 I/O 操作），CPU 转向另一个作业

多程序系统内存布局



现代操作系统的特征

- **多任务 (Multitasking)，分时系统 (Timesharing)**
- 进一步扩展多程序思想，使 CPU 更快速地在作业之间切换。这样，作业总能及时地得到 CPU，响应用户的交互操作，称为交互式计算。
  - **响应时间 (Response time)** 必须在 1 秒之内
  - 每个用户至少有一道作业在内存中执行 ⇨由此产生了进程
  - 如果存在两个以上的进程等待 CPU 执行 ⇨ 需要 CPU 调度
  - 如果内存空间装不下进程 ⇨ 需要换入、换出操作
  - 虚拟内存管理技术 使得小内存也能运行大进程

操作系统若干操作特征

- 中断驱动的硬件操作
- 软件申请，软件操作错误等，将产生异常，或陷入
- 面临
- “ 无限循环” 问题
- 进程干扰其它进程问题
- 进程干扰 OS 问题
- 等等

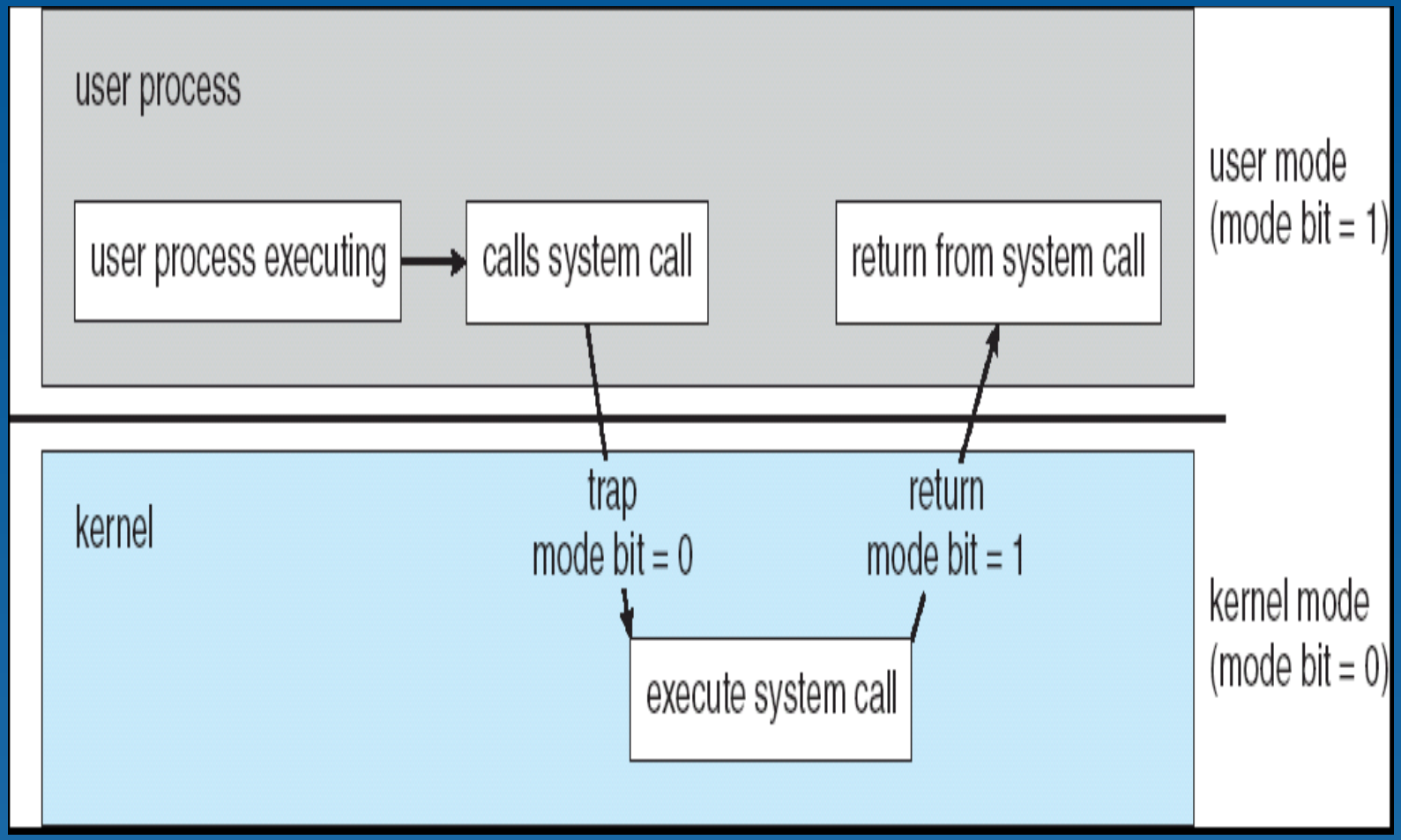
定时器 (Timer) 用来防止无限循环

- 设置定时中断
- 定时时间到，计数器做减 1 操作
- 计数器减至 0 ，触发一次中断
- 这种机制用来重获 CPU 控制权，或者终止一个程序

操作系统若干操作特征

- CPU 提供 **Dual-mode** 机制，实现 OS 自我保护
  - 用户态 (User mode) 和内核态 (kernel mode)
  - CPU 的 **Mode bit** 或者类似手段，可以在内核态和用户态之间切换
    - 以此区分系统在执行用户代码还是内核代码
    - 有些 CPU 带有特权指令，这些指令只能在内核态执行
    - 系统调用自动地从用户态切换至内核态，系统调用返回指令自动地从内核态切换至用户态

示例：从用户态切换至内核态



操作系统的服务类别

- 一类服务直接帮助用户
  - 用户界面 (UI) – 常见 UI 类别含
  - Command-Line (CLI)
  - Graphics User Interface (GUI)
  - 批处理 (Batch)
  - 程序执行 – 使 OS 能够装入程序到内存，执行驻留内存的程序，结束程序的执行，以及出错时的异常处理
  - I/O 操作

操作系统的服务类别（续）

- 一类服务直接帮助用户
  - 文件系统相关操作
  - 进程间通信
    - 通过共享内存实现通信
    - 通过消息传递实现通信
  - 出错检测 – OS 必须随时应对系统出错
    - 出错可能由硬件引起，如 CPU 、内存、 I/O 设备
    - 对于各种出错， OS 必须有合适的处理程序
    - OS 应该提供调试、查错工具

操作系统的服务类别（续）

- 另一类服务确保系统共享资源的高效运作
  - **资源分配**
  - **记账** – 跟踪记录哪些用户使用了多少资源，使用了哪些资源
  - **保护和安全**
    - **保护** – 确保对资源的访问都是受控的
    - **安全** – 外来访问都需通过身份认证，不允许非法访问

操作系统的服务功能

- 进程管理
- 内存管理
- 存储设备管理
- 大容量存储器管理
- I/O 子系统管理

操作系统用户界面 - CLI

- 命令行解释 (CLI) 直接解释执行命令项
  - 可以在 OS 内核实现，可以由系统程序实现
  - 可以有多个版本供用户选择 – **shells**
  - 基本流程：逐条取得用户的命令，逐条执行

操作系统用户界面 - GUI

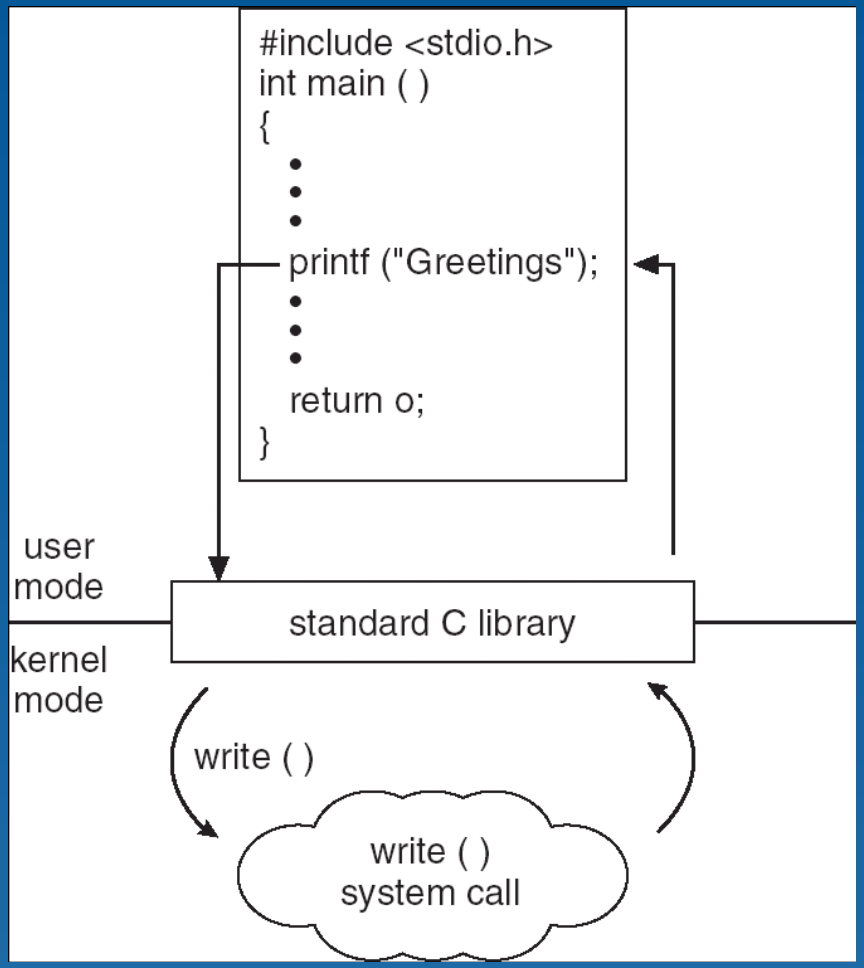
- 用户友善的桌面计算机系统操作界面
  - 硬件：鼠标、键盘、显示器
  - 以图标 **(Icons)** 表示文件、程序等对象
  - 由 Xerox PARC 发明
- 通常，操作系统既有 CLI ，又有 GUI
  - Microsoft Windows 以 GUI 为主，也支持 CLI （就是 “ command” shell ）
  - Apple Mac OS X 以 “ Aqua” 作为 GUI ， 以 UNIX 的 shell 作为 CLI
  - Solaris 以 CLI 为主，也支持 GUI (Java Desktop 、 KDE 等)

操作系统编程界面 - System Calls

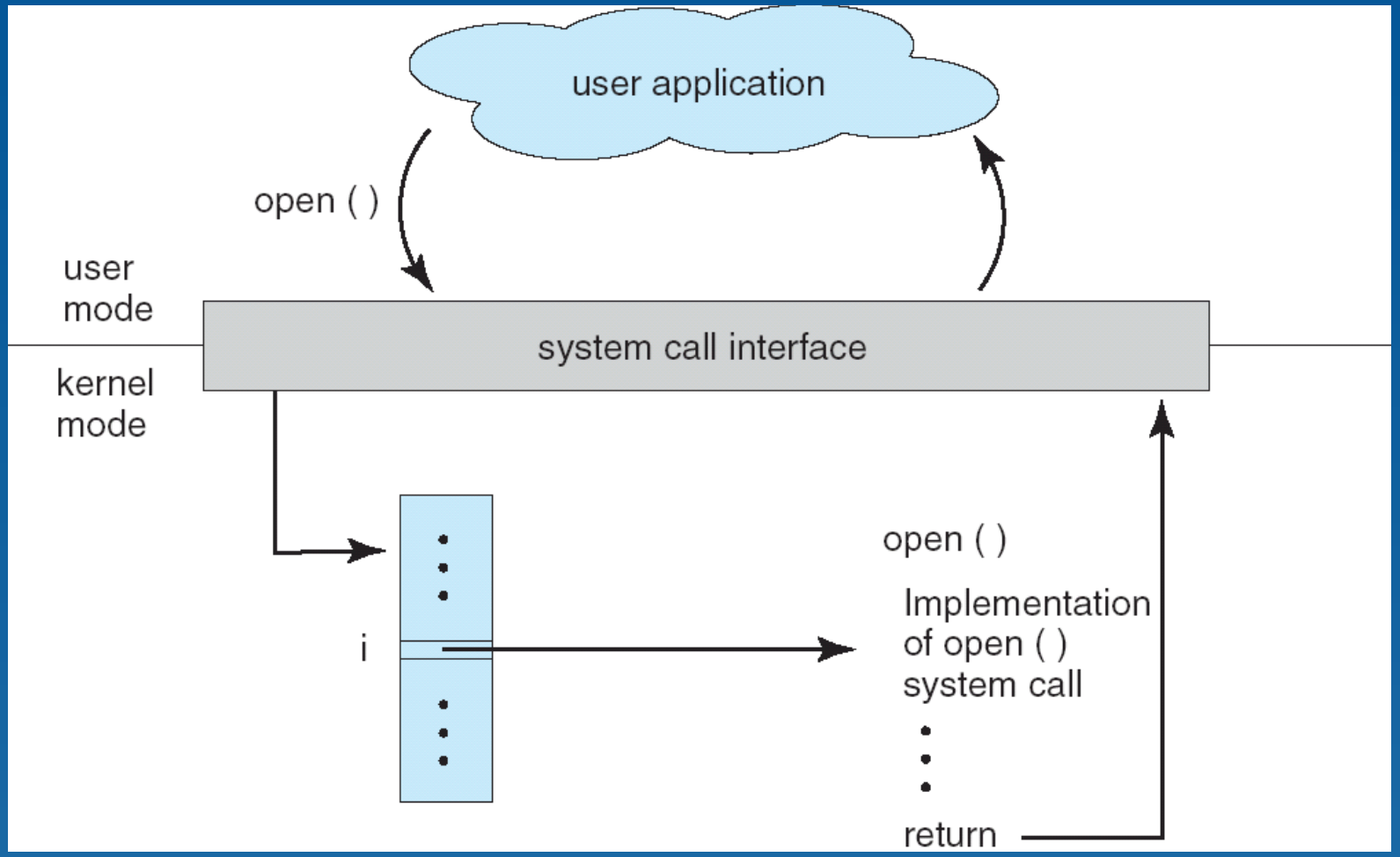
- 通常用高级语言 (C or C++) 实现
- 程序通常以 Application Program Interface (API) 使用，而不是直接使用系统调用
- 3 大流行的 APIs
- Win32 API for Windows
- POSIX API for POSIX-based systems (including virtually all versions of UNIX, Linux, and Mac OS X)
- Java API for the Java virtual machine (JVM)
- 为什么多数用 APIs，而不是 system calls？

示例：标准的 C 程序库

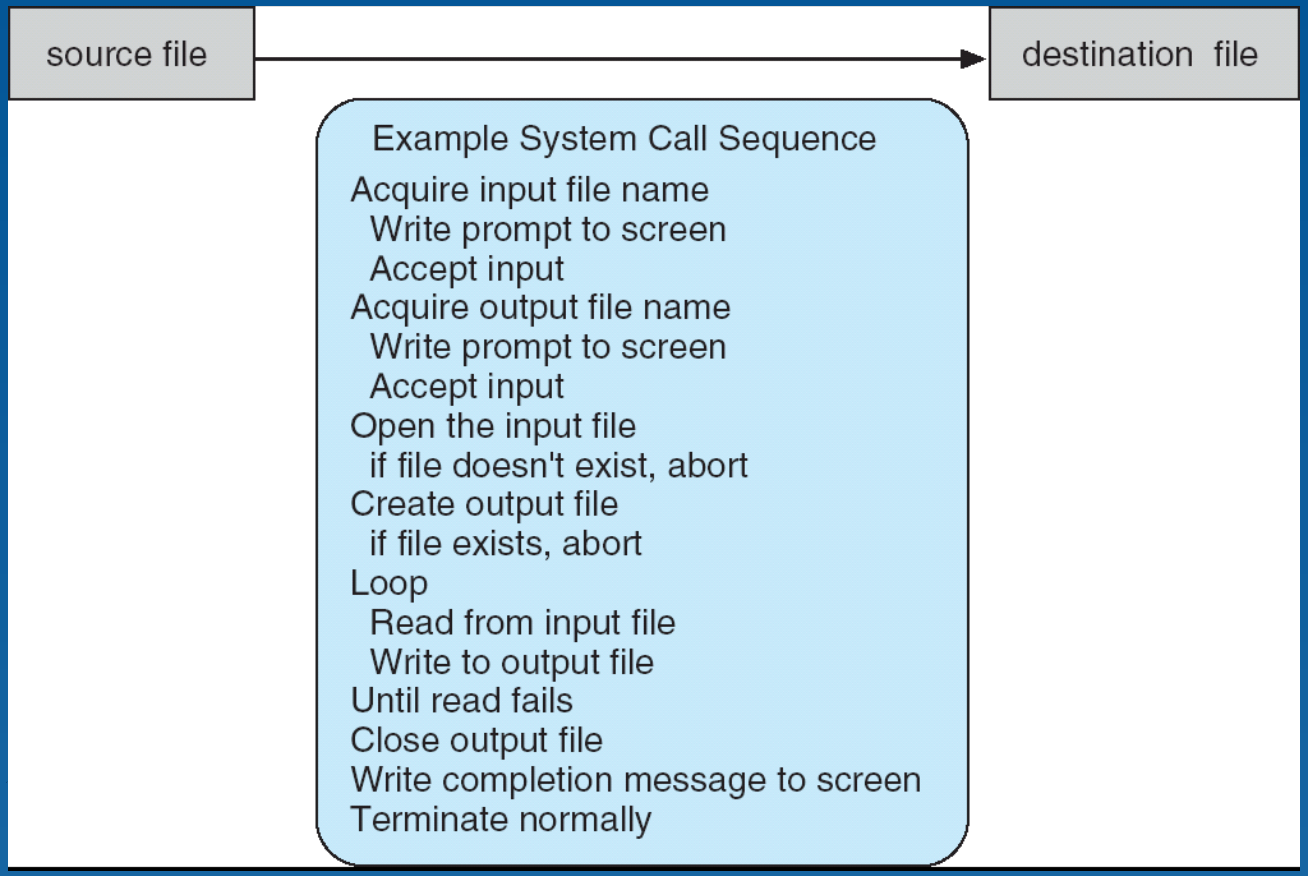
- C program invoking printf() library call, which calls write() system call



API – System Call – OS 之间的关联



示例：调 System Calls 进行文件复制



系统程序 (System Programs)

- 系统程序提供一套便捷的环境，利于程序开发和执行
- 系统程序分类
  - 文件操作
  - 状态信息展示
  - 文件内容修改
  - 编程语言支持
  - 程序装入和执行
  - 用户间通信
- 以用户考察 OS 的视角， OS 是以系统程序描述的，而不是系统调用

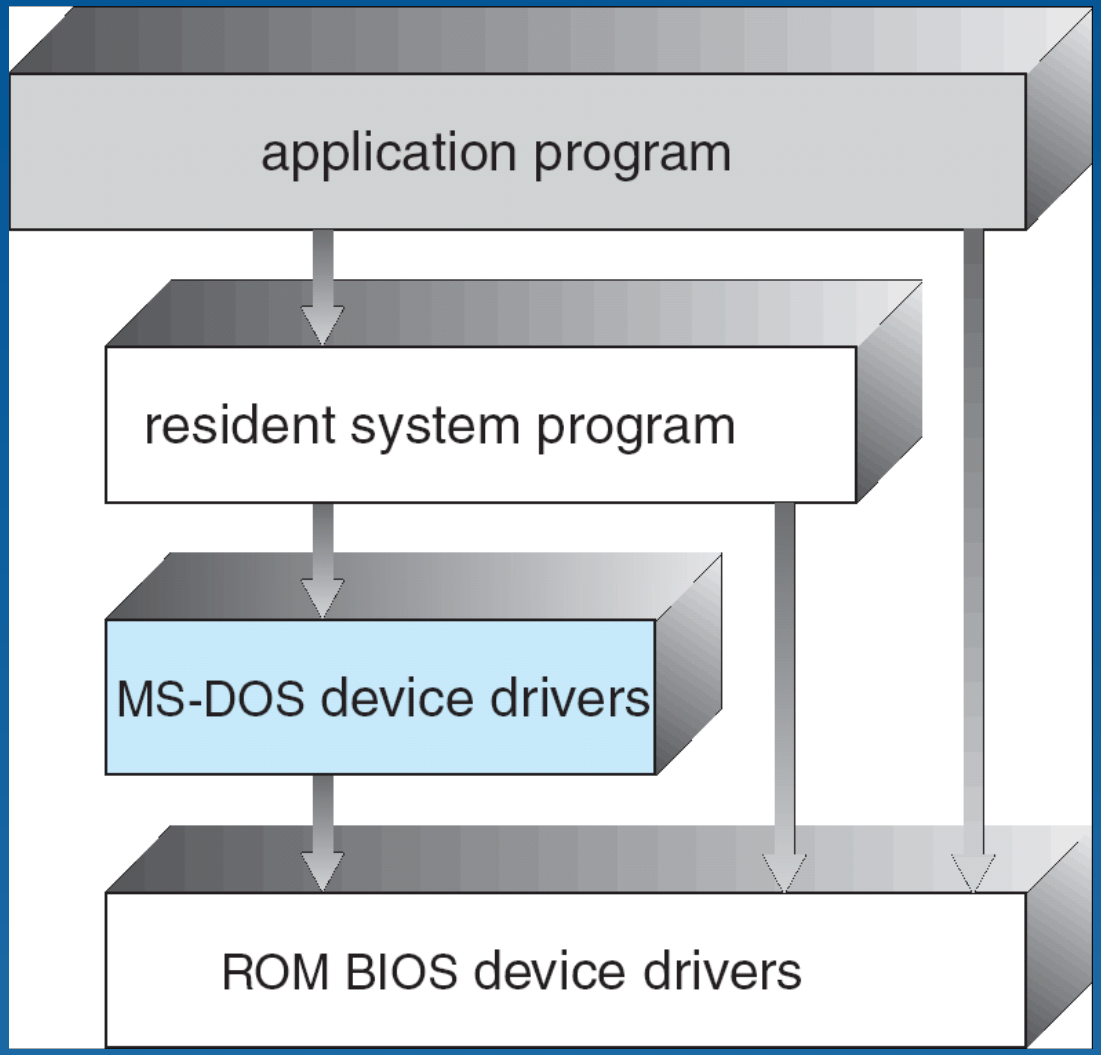
操作系统结构

- 简单结构
- 层次化方法
- 微内核结构
- 模块 (Modules)

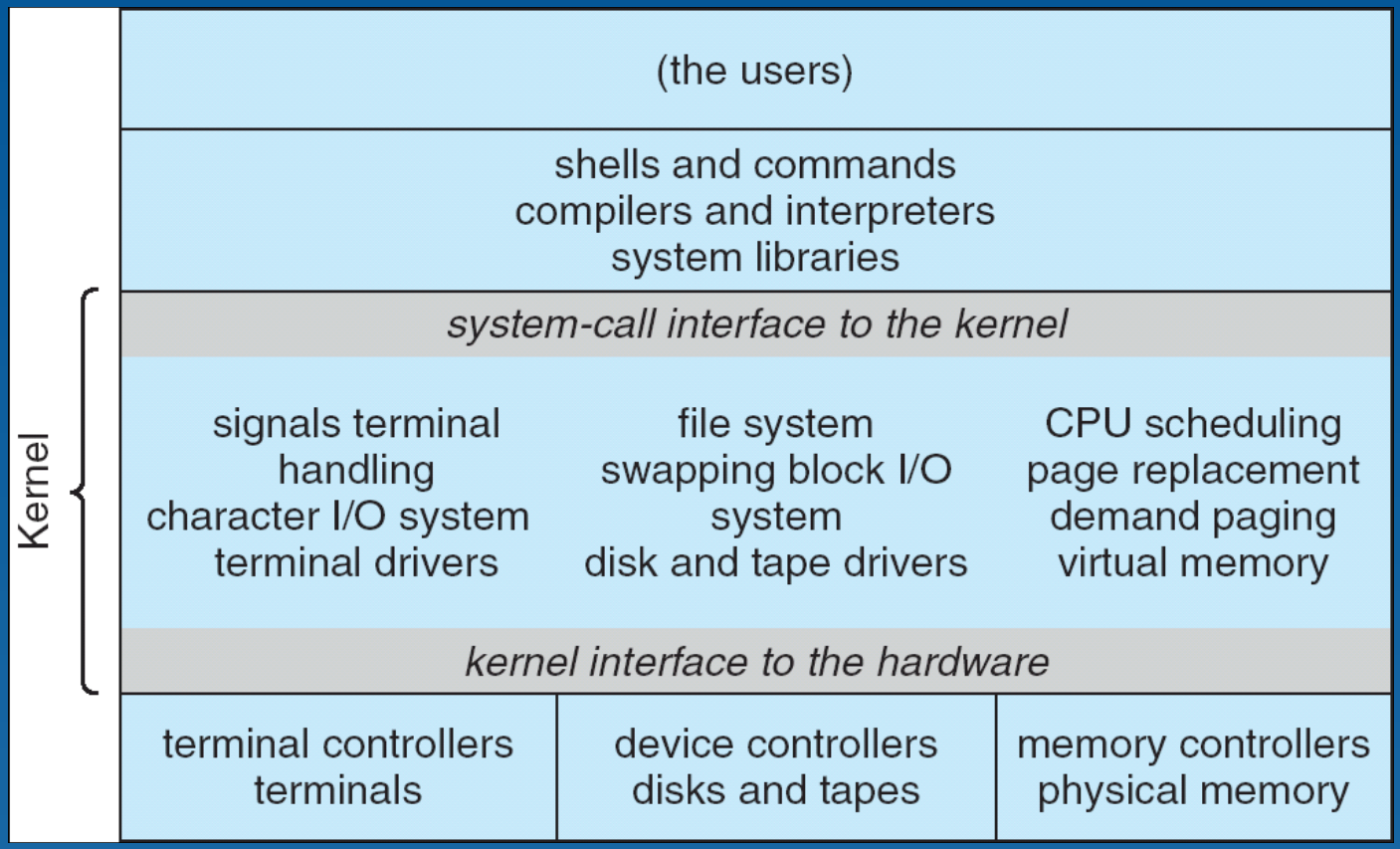
简单结构

- 以 MS-DOS 为代表
- 占用极小的内存空间，提供大部分 OS 功能
  - 不区分模块
  - 有一些数据结构。但是并没有很好地分离界面，层次化组织 OS 功能

MS-DOS 的层次结构


application program
resident system program
MS-DOS device drivers
ROM BIOS device drivers

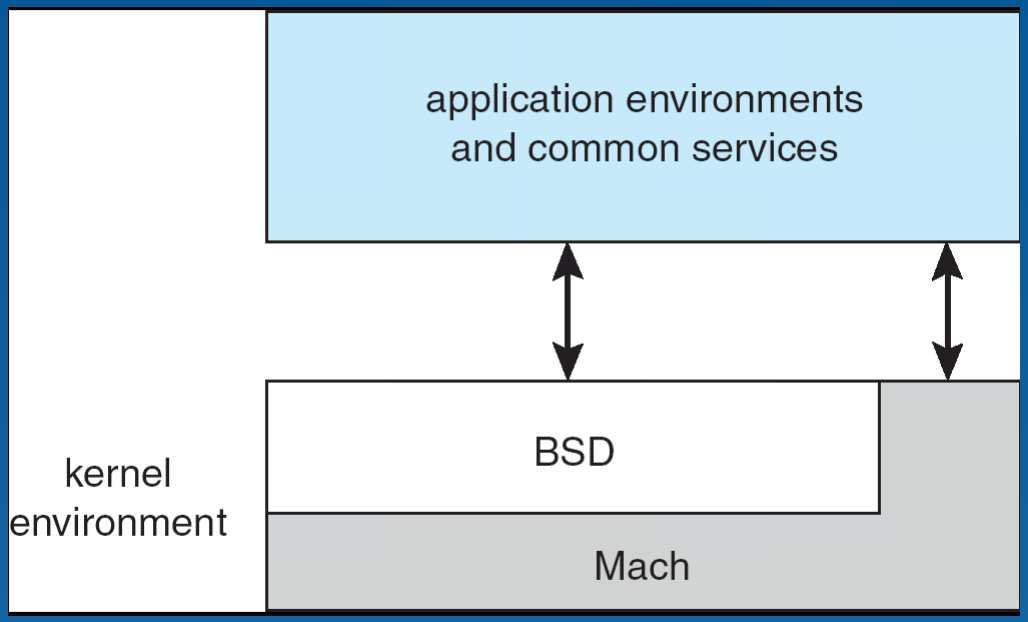
层次化方法，以 UNIX 为例



微内核结构

- 将 OS 的功能模块转移至用户态空间
- 剩下的，就是微内核
- 处于用户态空间的功能模块通过消息传递机制进行通信
- 有利因素
  - 容易升级微内核
  - 容易移植 OS 至不同类型的 CPU 、体系结构
  - 更可靠 (less code is running in kernel mode)
  - 更安全
- 不利因素
  - 用户态空间与内核态空间之间的通信频繁，性能开销大

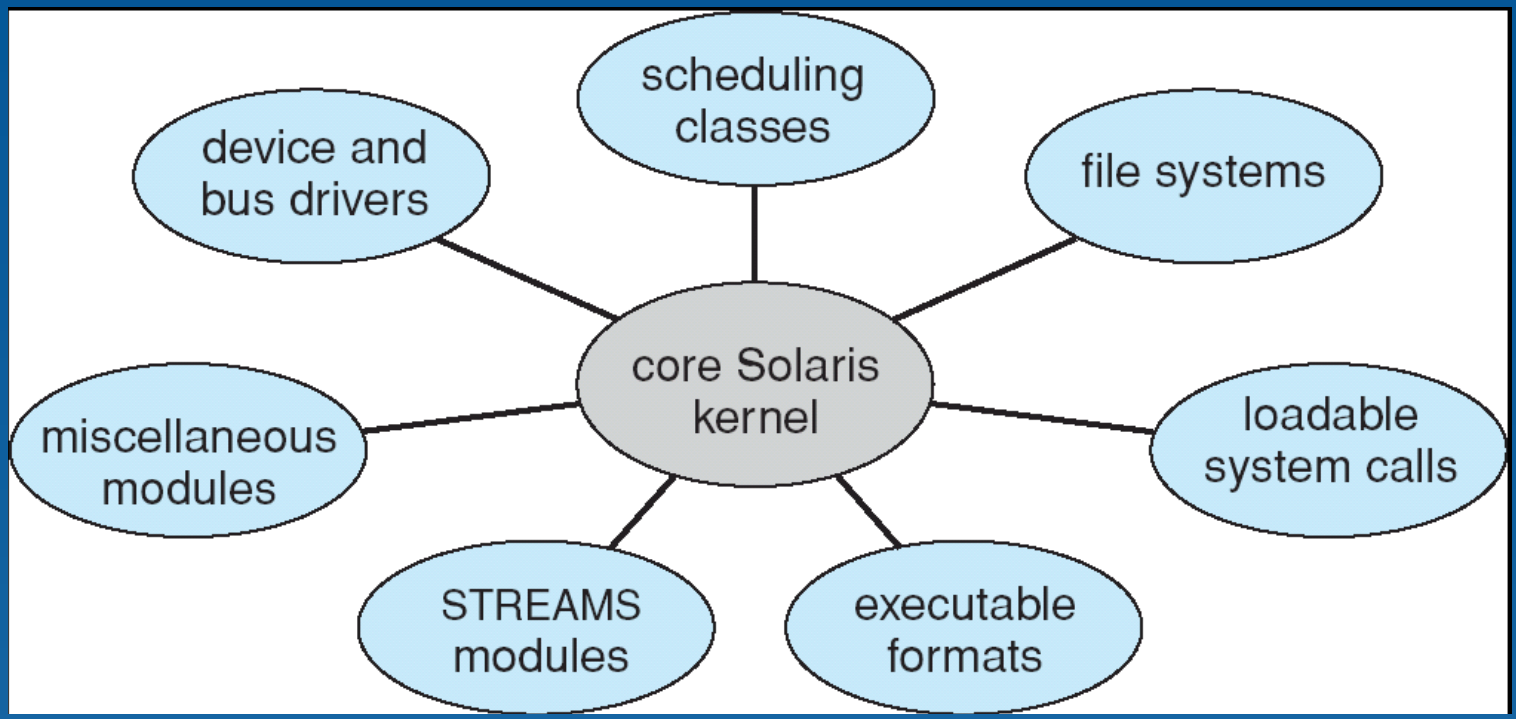
示例： Mac OS X


application environments
and common services
BSD
kernel
environment
Mach

模块 (Modules)

- 现代操作系统大多实现了内核模块 (kernel modules) 机制
  - 应用 object-oriented 思想方法
  - 核心组件相对独立、分离
  - 模块之间通过预知的界面对话
  - 可动态装入内核
  - 可动态卸载

示例： Solaris 的模块化方法




END