1.4 操作系统结构和运行模型

1.4.1 操作系统内核

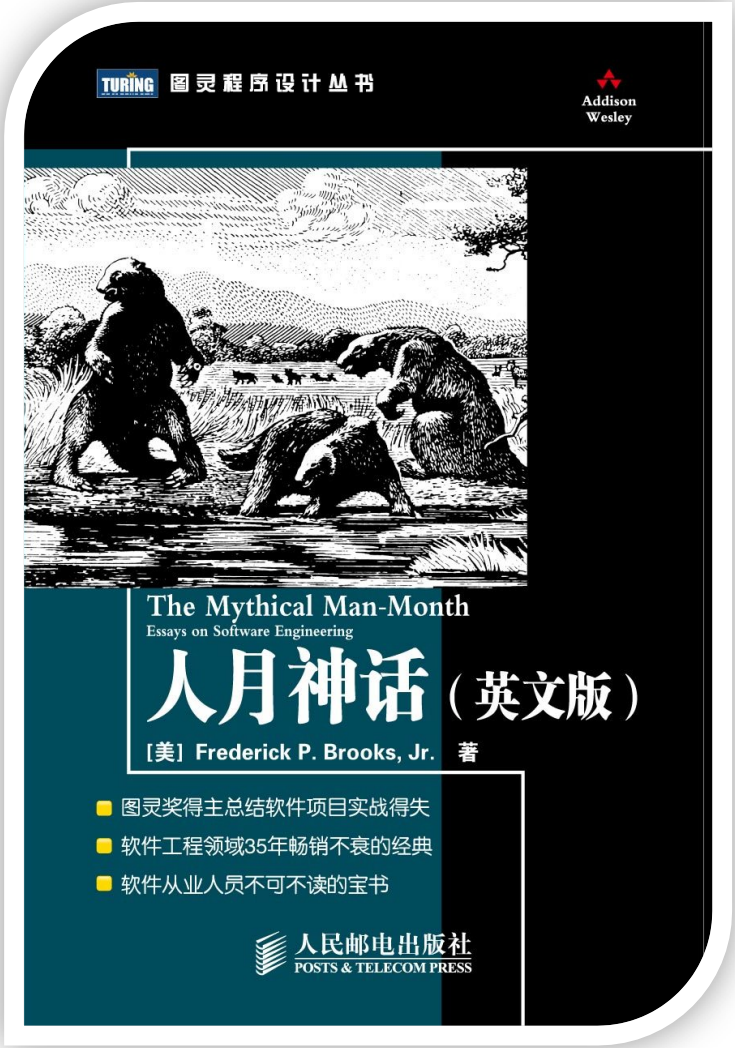
1.4.2 操作系统内核结构设计

1.4.3 操作系统运行模型

1.4 操作系统结构和运行模型

- 操作系统设计呈现出以下特征：
  1. 复杂程度高
  2. 生成周期长
  3. 正确性难保证
- 结构设计是操作系统开发的关键

- Windows 2000 超过3200万行语句，有2500个主要开发人员参与开发
- Linux 内核 拥有3.7万个文件，1500万行语句，主要参与人员达1300人（截至2011年）
- Android 4.0 包含100万行语句 （截至2011年）



1.4 操作系统结构和运行模型

- 一个OS开发完成后，还需大量时间开发支持该系统的大量应用程序；
  - iOS, Android, Linux, …
- 待应用程序问世后，用户还必须通过培训及实践去学会操作和使用；
- 这意味着用户拥有并使用的是10年或20年前的操作系统技术；
- 而计算机硬件却在不断地更新换代，刚刚开发完成的操作系统又需要升级。

需要一个稳健、高效、安全的操作系统架构

1.4 操作系统结构和运行模型

- 操作系统在结构设计上提出了一系列重要概念
  - 模块化、层次化、虚拟化
- 操作系统构件包括：
  - 内核、进程、线程等
- 内核设计是操作系统设计中最为重杂的部分
  - 支撑软件、应用软件等上层软件离不开内核的支撑

1.4 操作系统结构和运行模型

- 操作系统结构设计有三层含义
  1. 研究操作系统的整体结构 （*1.4.1*）
     - 功能如何分块、模块间如何交互、构造的过程和方法
  2. 研究操作系统程序的局部结构（*1.4.3*）
     - 数据结构、控制结构
  3. 研究操作系统运行模型（*1.4.4*）
     - 操作系统服务例程与应用进程如何交互、运行

1.4 操作系统结构和运行模型

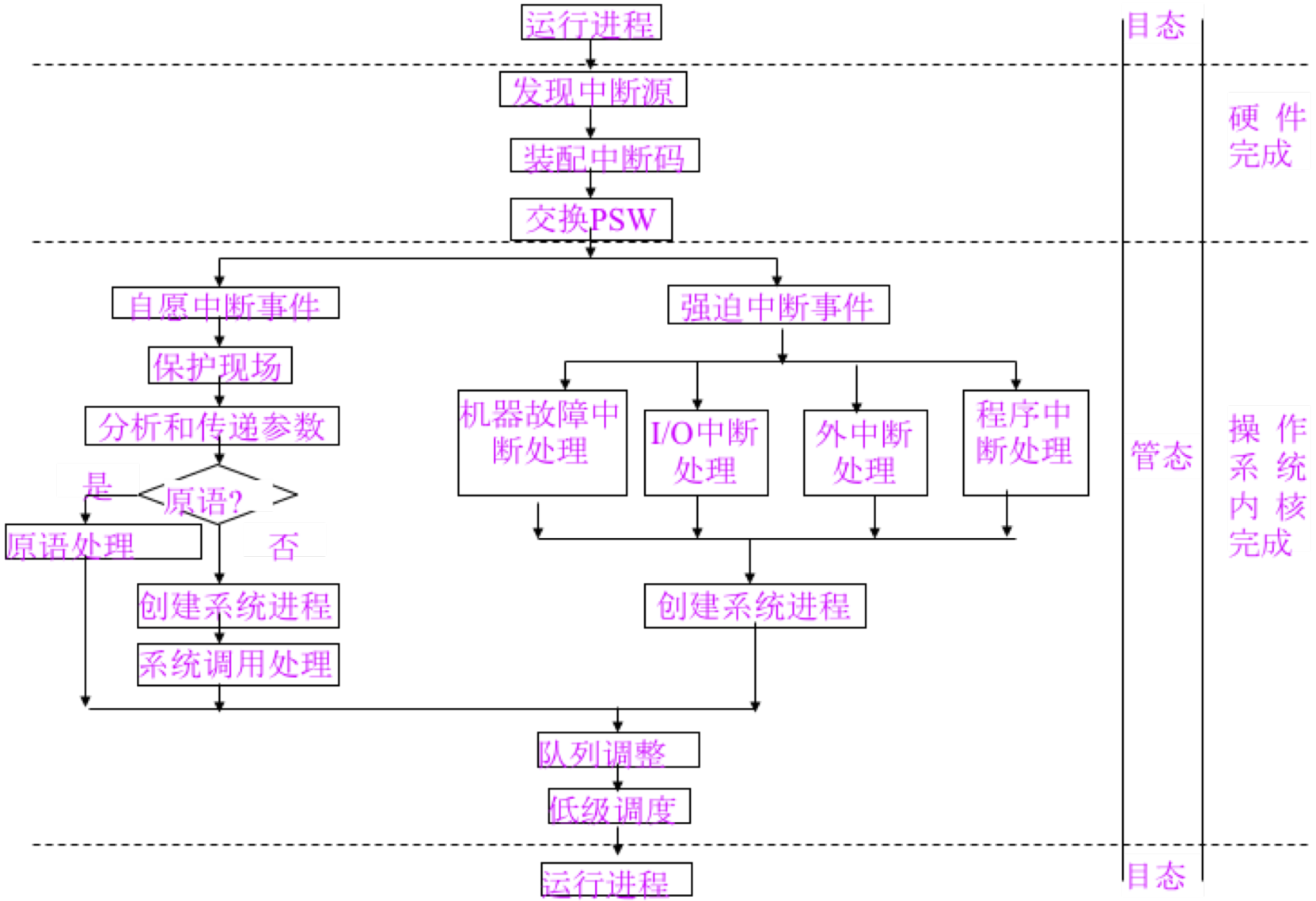
1.4.1 操作系统内核

1.4.2 操作系统内核结构设计

1.4.3 操作系统运行模型

1.4.1 操作系统内核

- 内核不是进程，而是支持系统运行基本功能的一组程序模块。内核对硬件处理器及有关资源进行首次改造，以便给进程的执行提供良好的运行环境
- 内核分为：
  - 单内核：内核具有较多的功能，运行时是一个大的二进制映像，模块间的联系通过函数或过程调用实现
  - 微内核：内核很小，仅具有极少的必须功能，其它功能都在核外实现。通过微内核提供的消息传递机制完成其余功能模块间的联系。内核和核外服务程序的开发是分离的



内核被触发和内核处理流程

1.4.1 操作系统内核

- 内核的功能
  - 中断处理：
    - 为了缩短屏蔽中断的时间，增加系统内的并发性，内核通常仅进行有限的、简短的处理：分析中断事件的类型和性质，进行必要的状态修改，然后交给内核之外的（可中断）进程去处理
  - 时钟管理：时间片调度，定时唤醒等
  - 短程调度：主要职能是分配处理器，处理器转让，保护恢复现场等
  - 原语管理：
    - 原语是内核中实现某一功能的不可中断过程
    - 原语由内核完成，系统调用由系统进程实现

1.4.1 操作系统内核

- 内核的属性
  1. 由中断驱动
     - 只有当发生中断事件后由硬件交换程序状态字才引出操作系统的内核进行中断处理，且在处理完中断事件后内核自行退出
  2. 内核的执行是连续的
     - 在内核执行期间不能插入内核以外的程序执行
     - 可陷入到高级中断
  3. 可以在屏蔽中断状态下执行
     - 在处理某个中断时，为避免中断的嵌套可能引起的错误，必须屏蔽该级中断
  4. 可以使用特权指令
     - 现代计算机都提供常态和特态等多种机器工作状态，有一类指令称为特权指令，只允许在特态下使用，规定这类指令只允许内核使用，可防止系统出现混乱

1.4 操作系统结构和运行模型

1.4.1 操作系统内核

1.4.2 操作系统内核结构设计

1.4.3 操作系统运行模型

1.4.2 操作系统内核结构设计

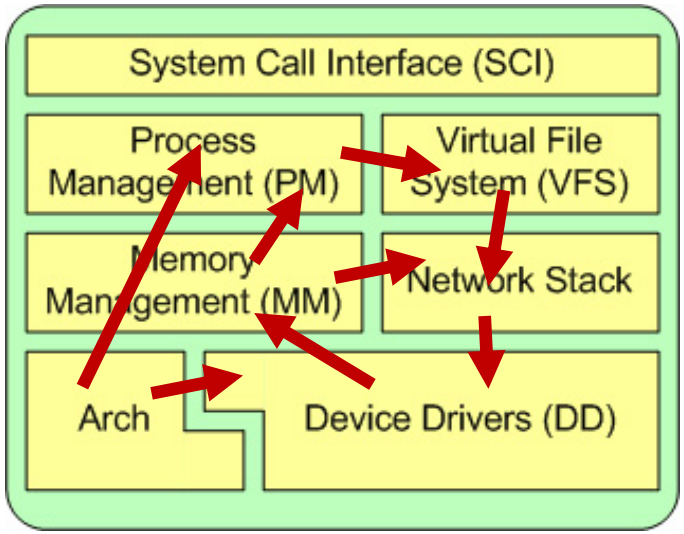
- 单内核：起始于 1960 年代，内核中各部件杂然混居，被 Windows, Unix/Linux 广泛采用；
- 微内核：起始于 1980 年代，强调结构性部件与功能性部件的分离，产品化很少；
- 混合内核：微内核和单内核的折中，较多组件在核心态中运行，自 Windows NT 开始被应用，受微内核性能影响，其设计理念争议较大；
- 外内核：多应用于嵌入式系统，偏重于硬件抽象化

1.4.2 操作系统内核结构设计

- 单体式结构
  - 以功能模块为操作系统的基本单位
    - 每个模块有一定的独立功能，若干模块可协作完成某个功能
    - 模块间调用方法自由、随意，可不加控制
    - 模块分别设计、编码、调试

- 结构紧密
- 组合方便
- 灵活性高
- 系统效率高

- 模块独立性差
- 模块间调用关系复杂
- 系统结构不清晰
- 难以保证系统正确性

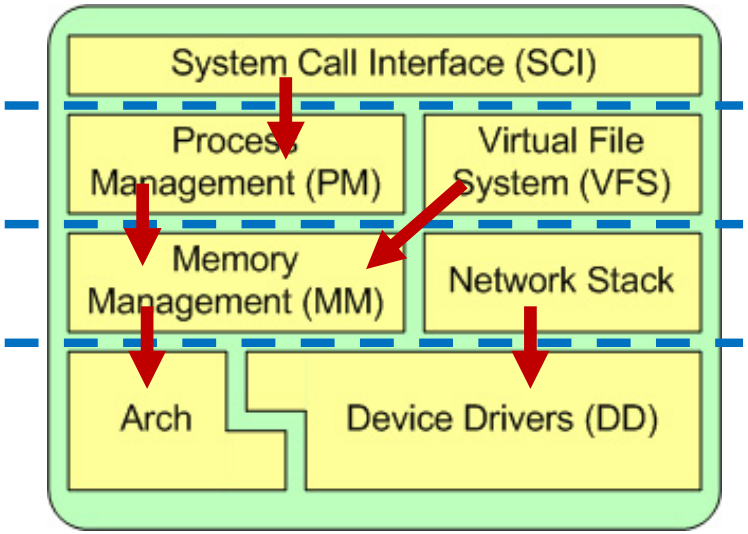


1.4.2 操作系统内核结构设计

- 层次式结构
  - 把操作系统划分为内核和模块（进程）
  - 模块（进程）按功能调用次序排列成层次
    - 层次间只能单向依赖和调用，高层可调用低层功能，反之不能
    - 层次间不构成循环调用关系

- 规范层次依赖和调用关系
- 系统接口少、简单
- 逐层保证正确性

- 层次间通信开销较大
- 影响系统效率



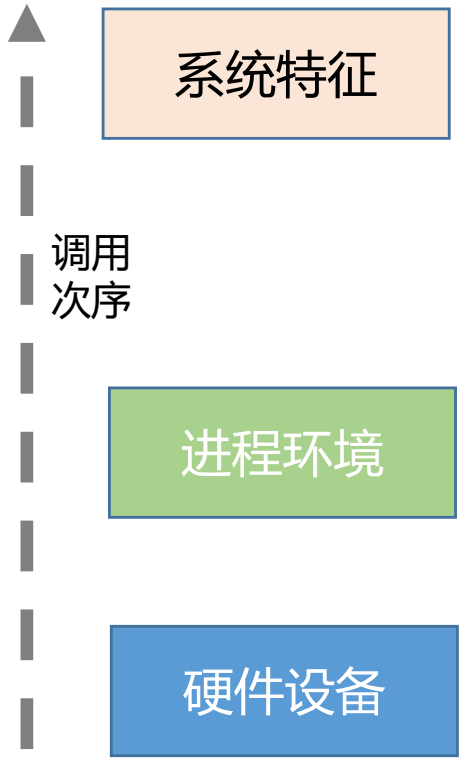
1.4.2 操作系统内核结构设计

- 层次式结构构造方法
  - 自底向上方法
    - 从裸机开始，逐步添加各层软件，形成接近目标虚拟机的系统
  - 自顶向下方法
    - 自顶向下方法从目标系统出发，通过若干层软件过渡到宿主机器，其实质是对目标系统的逐步求精

THE 多道程序设计系统，1968

1.4.2 操作系统内核结构设计

- 层次式结构分层原则
  - 把与机器硬件有关的程序模块放在最底层，以便把硬件与其它层隔离开来
  - 为进程（线程）的正常运行创造环境和提供条件的内核程序应该尽可能放在底层
  - 反映系统外特性的软件放在最外层
  - 按照实现操作系统命令时模块间的调用次序或按进程间单向发送信息的顺序来分层

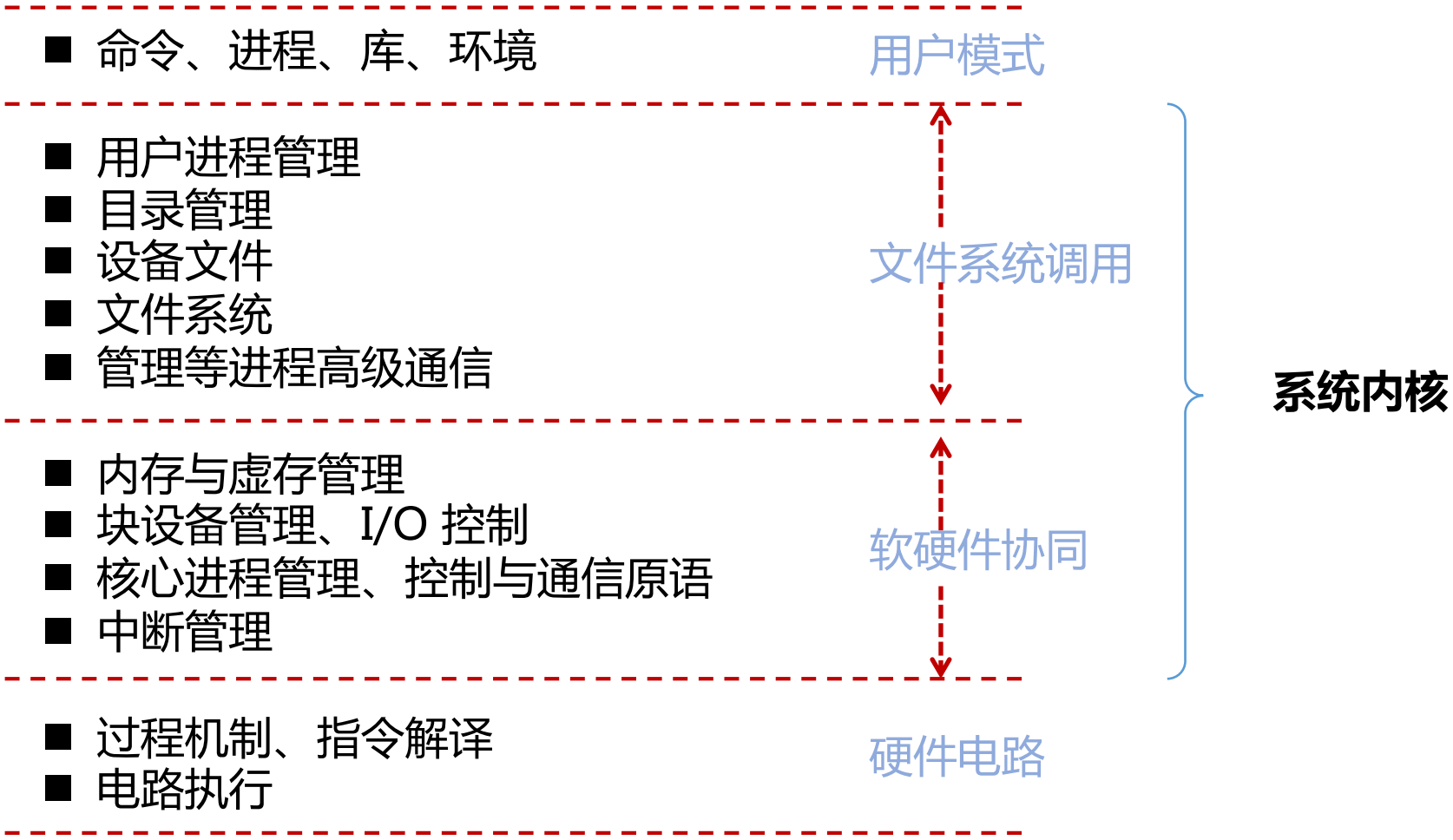


1.4.2 操作系统内核结构设计

- 层次式结构设计示例（粗粒度划分）

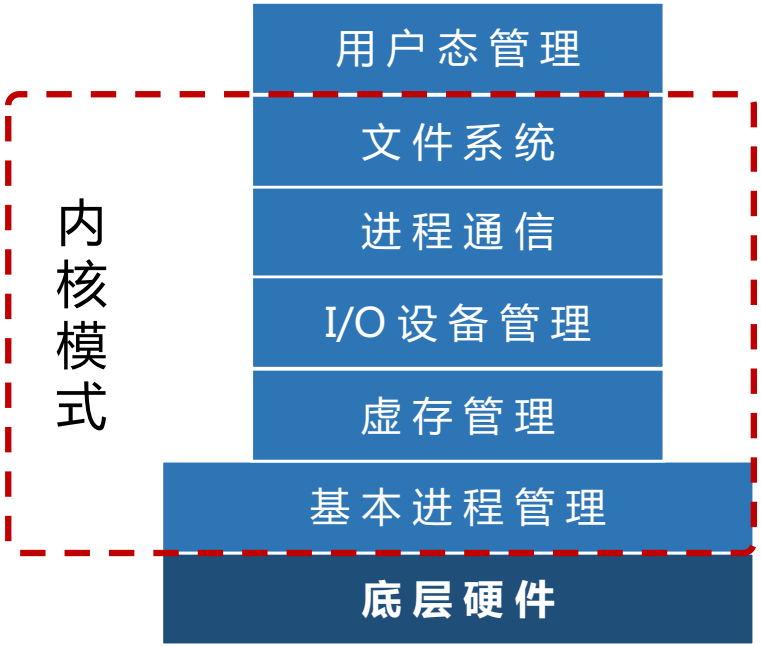
1.4.2 操作系统内核结构设计

- 层次式结构设计示例（细粒度展示）

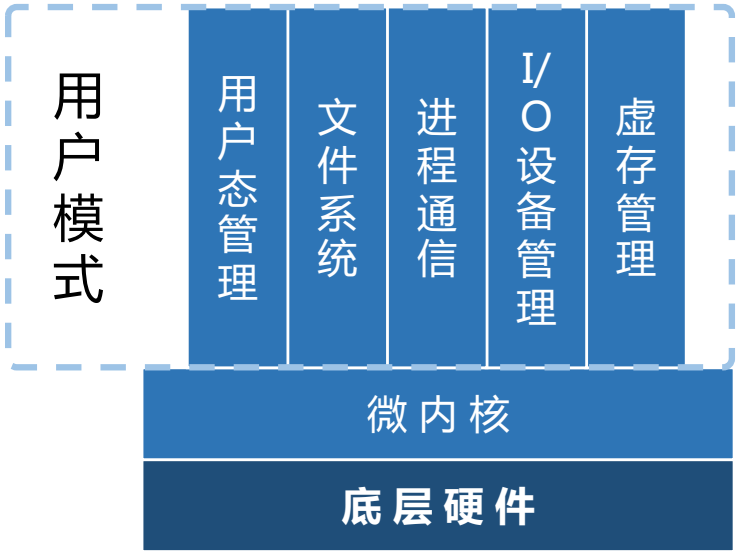


1.4.2 操作系统内核结构设计

- 微内核结构
  - 封装所有应用必需的核心功能，形成微内核（microkernel）
  - 其它功能形成运行在用户态的服务进程。



（a）分层单内核结构



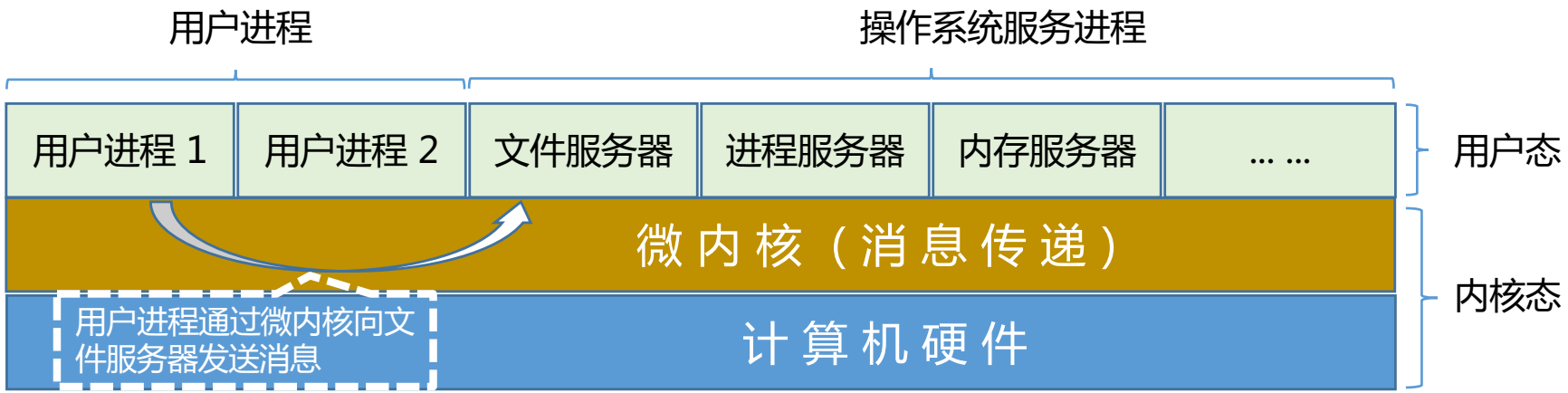
（b）微内核结构

1.4.2 操作系统内核结构设计

- 微内核结构
  - 将操作系统分成两大部分
    1. 运行在用户态并以客户/服务器方式活动的进程
    2. 运行在核心态的内核
  - 除内核部分外，操作系统的其他部分被分成若干个相对独立的进程，每一个进程实现一类服务，称服务器进程

1.4.2 操作系统内核结构设计

- 微内核结构
  - 客户进程发出消息，内核将消息传送给服务器进程，服务器进程执行客户提出的服务请求，并通过内核发送消息将结果返回给客户



1.4.2 操作系统内核结构设计

- 微内核结构
  - 微内核只实现极少任务，特别是那些直接依赖于硬件的功能
    - 进程通信
    - 少量内存管理
    - I/O和中断管理
  - 主要起到信息验证、消息交换的作用
  - 这种结构也被称为客户/服务器与微内核结构

1.4.2 操作系统内核结构设计

- 微内核结构
  - 微内核只实现极少任务，特别是那些直接依赖于硬件的功能
    - 进程通信
    - 少量内存管理
    - I/O和中断管理

例：对存储管理而言，为了实现进程级的保护，微内核必须控制地址空间的硬件设施，因此：

- 内核负责把每个虚页面映射到物理页框
- 大量的存储管理功能，包括进程地址空间之间的相互保护、页面淘汰算法等功能可在内核外实现

1.4.2 操作系统内核结构设计

- 微内核结构

- 提供一致性接口，不用区分内核级服务还是用户级服务
- 良好的可扩充性，增加新功能只需更新服务器即可
- 可移植性好，与CPU相关的代码都在微内核中
- 支持分布式系统，只需要发布消息，不必知道服务器是在哪台硬件设备上

- 所有进程通信，都必须通过内核的通信机制才能进行通信
- 系统效率低

1.4 操作系统结构和运行模型

1.4.1 操作系统内核

1.4.2 操作系统内核结构设计

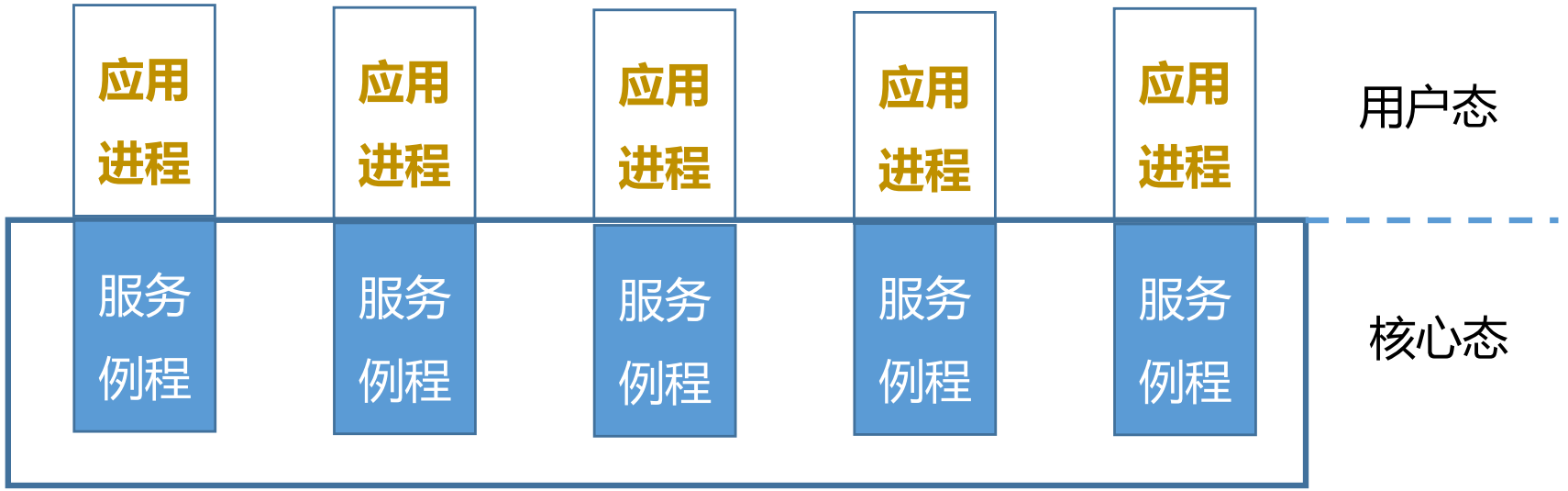
1.4.3 操作系统运行模型

1.4.3 操作系统运行模型

- 操作系统服务例程与应用进程如何交互、运行
  - 两个问题：
    1. 操作系统组成子程序为用户调用，还是组成进程提供服务
    2. 操作系统程序运行在系统空间，还是用户空间
  - 两种模型：
    1. 子程序：服务例程嵌入到应用进程中运行，伴随在整个应用程序的生命周期内；
    2. 进程：服务例程作为独立进程运行

1.4.3 操作系统运行模型

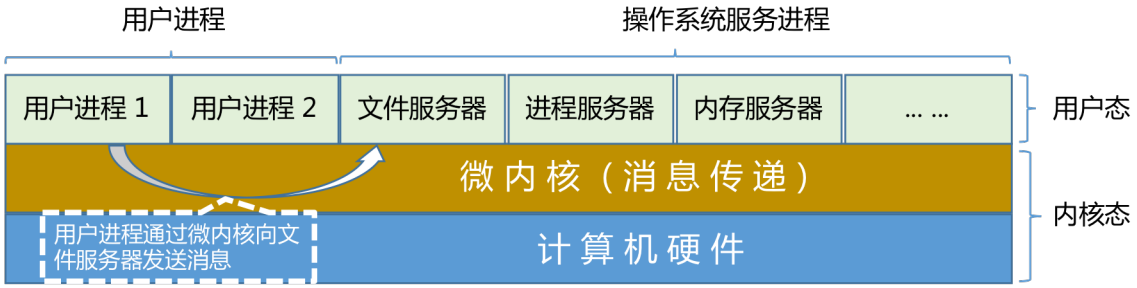
1. 服务例程嵌入到应用进程中运行
   - 操作系统的地址空间被包含在用户进程的地址空间中
   - 操作系统例行程序也在用户进程的上下文环境中执行
   - 只切换处理器状态，而不切换进程，效率较高



1.4.3 操作系统运行模型

2. 服务例程作为独立进程运行

- 即“微内核”架构



- 采用了模块化的操作系统实现方法
- 操作系统功能被组织成独立的进程，有利于操作系统的实现、配置和扩充
- 在多处理器多计算机的环境下非常有效，一些系统服务可指派到专门处理器上执行

- 所有进程通信，都必须通过内核的通信机制才能进行通信
- 系统效率低