操作系统结构设计

Linux
Android
Linux
OpenStack
Mac OS
Windows

预备知识：程序是如何运行的？

C语言代码 → 编译器（“翻译”） → 机器指令（二进制）

一条高级语言的代码翻译过来可能会对应多条机器指令

```
Int x = 1;
x++;
```

→

```
10001010110000110001 1100
00101100000000 1101001111
10010010000000 1100000001
00101100010011 1100000011
```

程序运行的过程其实就是CPU执行一条一条的机器指令的过程

预备知识：程序是如何运行的？

C语言代码 → 编译器（“翻译”） → 机器指令（二进制）

一条高级语言的代码翻译过来可能会对应多条机器指令

Int x = 1;
x++;

→

```
10001010110000110001 1100
00101100000000 1101001111
10010010000000 1100000001
00101100010011 1100000011
```

程序运行的过程其实就是CPU执行一条一条的机器指令的过程

“指令”就是处理器（CPU）能识别、执行的最基本命令

注：很多人习惯把 Linux、Windows、MacOS 的“小黑框”中使用的命令也称为“指令”，其实这是“交互式命令接口”，注意与本节的“指令”区别开。本节中的“指令”指二进制机器指令

内核程序 v.s. 应用程序

C语言代码 → 编译器（“翻译”） → 机器指令（二进制）

一条高级语言的代码翻译过来可能会对应多条机器指令

```
Int x = 1;
x++;
```

```
10001010110000110001110 0
00101100000000110100111 1
10010010000000110000000 1
00101100010011110000001 1
```

程序运行的过程其实就是CPU执行一条一条的机器指令的过程

我们普通程序员写的程序就是“应用程序”

微软、苹果有一帮人负责实现操作系统，他们写的是“内核程序”
由很多内核程序组成了“操作系统内核”，或简称“内核（Kernel）”
内核是操作系统最重要最核心的部分，也是最接近硬件的部分
甚至可以说，一个操作系统只要有内核就够了（eg：Docker—>仅需Linux内核）
操作系统的功能未必都在内核中，如图形化用户界面 GUI

特权指令 v.s. 非特权指令

程序运行的过程其实就是CPU执行一条一条的机器指令的过程

应用程序只能使用“非特权指令”，如：加法指令、减法指令等

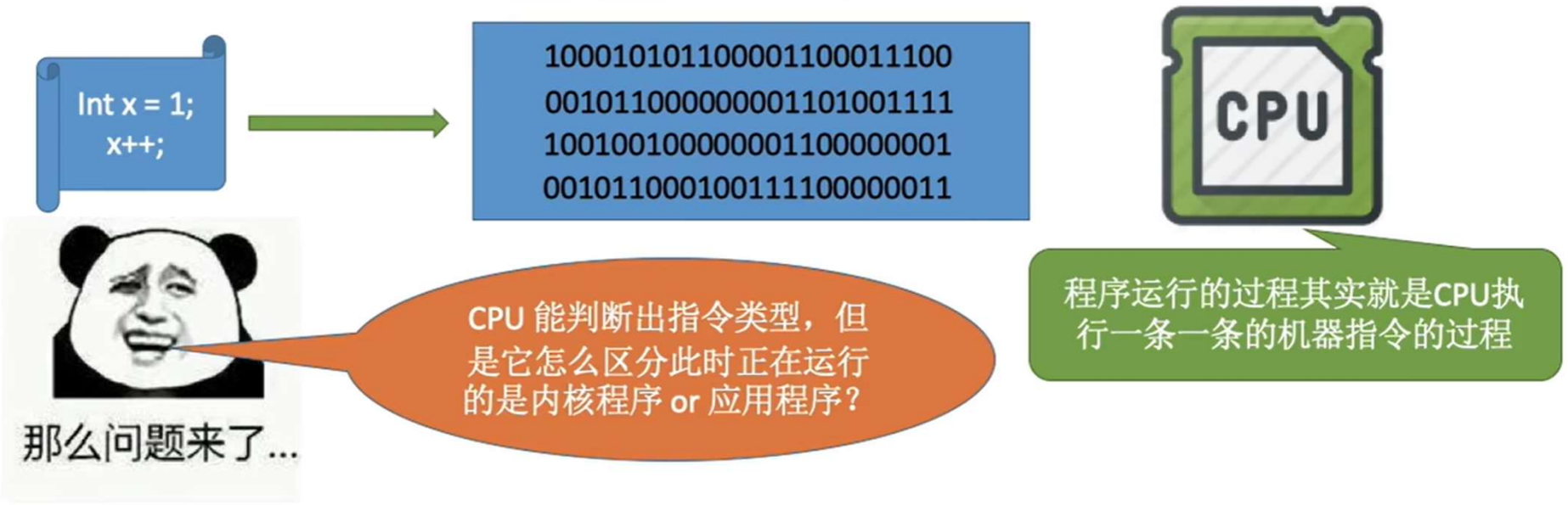
我们普通程序员写的程序就是“应用程序”

微软、苹果有一帮人负责实现操作系统，他们写的就是“内核程序”

操作系统内核作为“管理者”，有时会让CPU执行一些“特权指令”，如：内存清零指令。这些指令影响重大，只允许“管理者”——即操作系统内核来使用

在CPU设计和生产的时候就划分了特权指令和非特权指令，因此CPU执行一条指令前就能判断出其类型

内核态 v.s. 用户态



CPU 有两种状态，“内核态”和“用户态”
处于内核态时，说明此时正在运行的是内核程序，此时可以执行特权指令
处于用户态时，说明此时正在运行的是应用程序，此时只能执行非特权指令

拓展：CPU 中有一个寄存器叫 程序状态字寄存器（PSW），其中有个二进制位，1表示“内核态”，0表示“用户态”
别名：内核态=核心态=管态；用户态=目态

内核态 v.s. 用户态

操作系统的结构设计

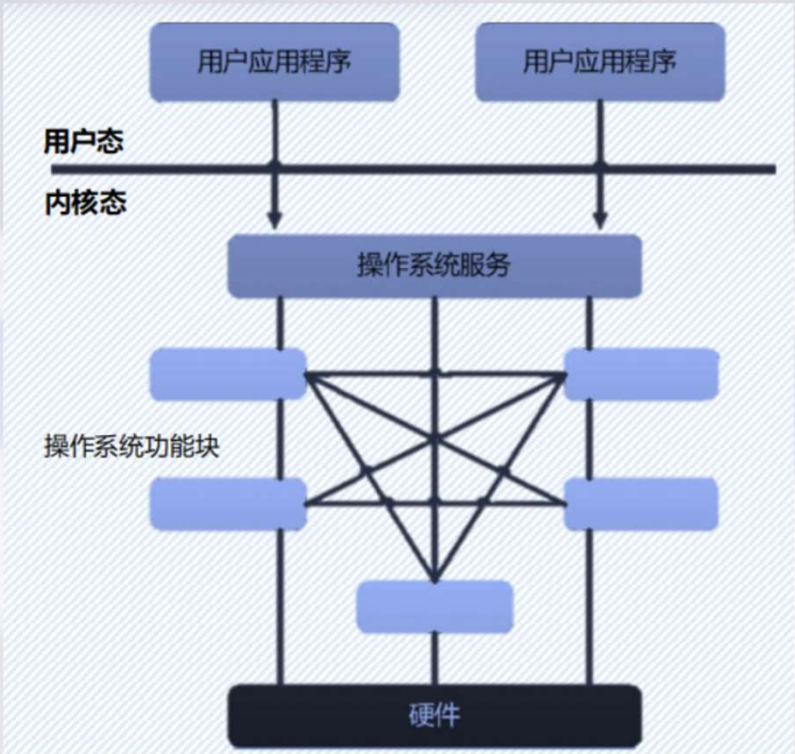
本讲内容

1. 整体式结构
2. 层次式结构
3. 虚拟机结构
4. 客户/服务器结构
5. 微内核结构

整体式结构的操作系统

定义

整体式结构又叫**模块组合法**，是基于结构化程序设计的软件结构设计方法。



操作系统分为：处理器管理模块，内存管理模块，IO设备管理模块，文件管理模块

整体式结构的操作系统

主要设计思想

- 将模块作为操作系统的基本组成单位
- 按照功能需要而不是根据程序和数据的特性把整个系统分解为若干模块
- 模块可以再进一步分成子模块
- 每个模块具有一定独立功能，多个模块协作完成某个功能
- 各模块分别设计、编码、调试
- 所有模块连结成一个完整的系统

整体式结构的操作系统

主要优点

- ✓ 结构紧密、组合方便，对不同环境和用户的不同需求，可以组合不同模块来满足，灵活性大(服务器/个人电脑)
- ✓ 每个功能可以用最有效的算法和调用其它模块中的过程来实现，系统效率较高
- ✓ 设计及编码可齐头并进，加快操作系统研制过程

整体式结构的操作系统

主要缺点

- ✓ 模块独立性差，模块之间牵连多(ABC模块循环调用，出现死循环)
- ✓ 形成了复杂的调用关系，甚至有循环调用
- ✓ 系统结构不清晰，正确性、可靠性降低
- ✓ 系统功能的增、删、改十分困难

操作系统的结构设计

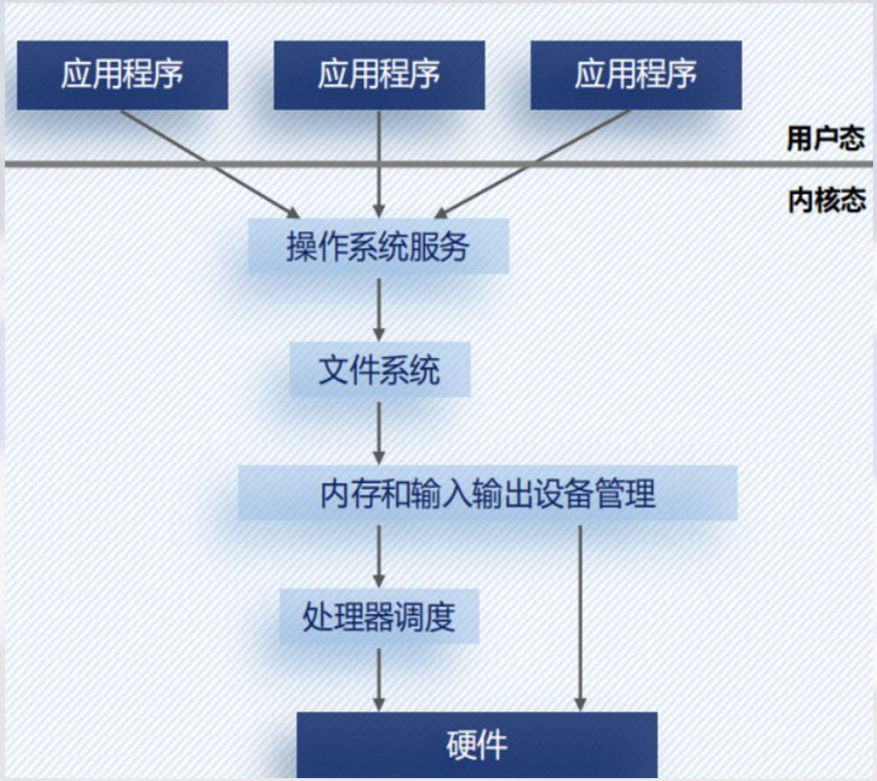
本讲内容

1. 整体式结构
2. 层次式结构
3. 虚拟机结构
4. 客户/服务器结构
5. 微内核结构

层次式结构的操作系统

层次结构

- 操作系统划分为内核和若干模块
- 模块按功能的调用次序排列成若干层次
- 各层之间只能是单向依赖或单向调用关系



层次式结构的操作系统

层次结构类型

单向依赖：上层可以调用下层，下层不能调用上层

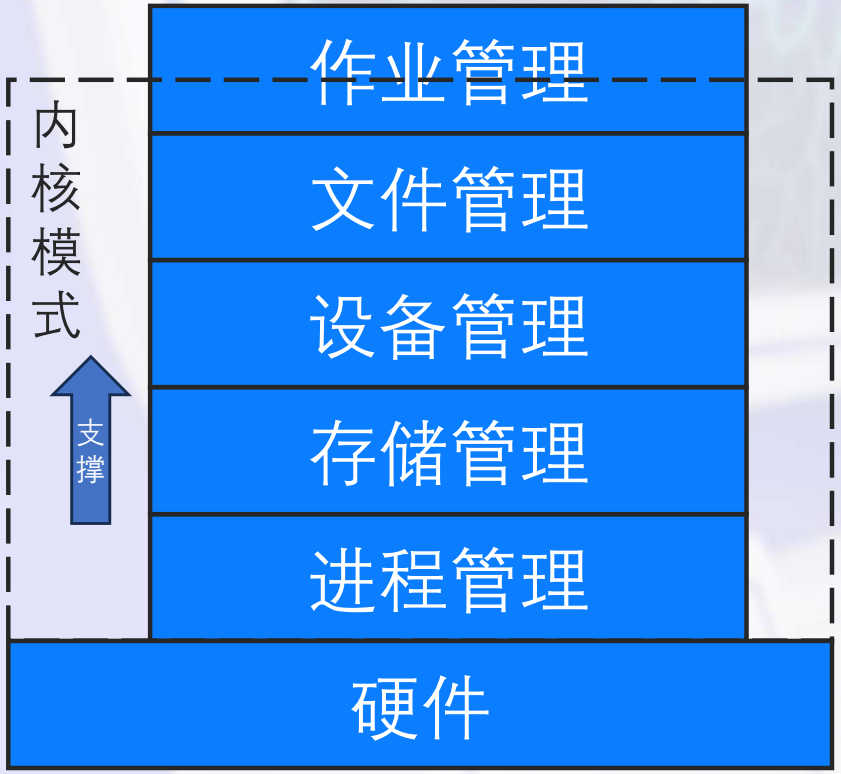
全序

- 各层之间是单向依赖的
- 层内模块之间也保持独立，没有联系

半序

- 各层之间是单向依赖的
- 层内允许有相互调用或通信的关系

层次式结构的操作系统

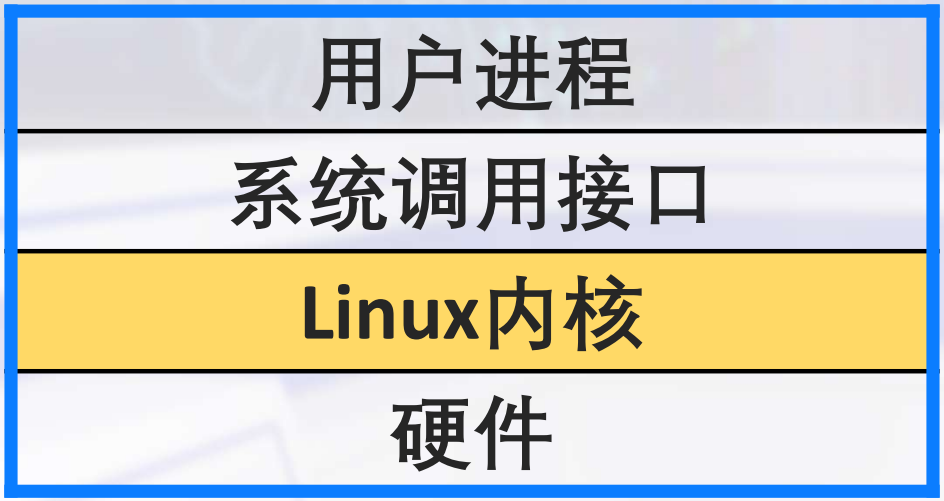


层次式结构优点

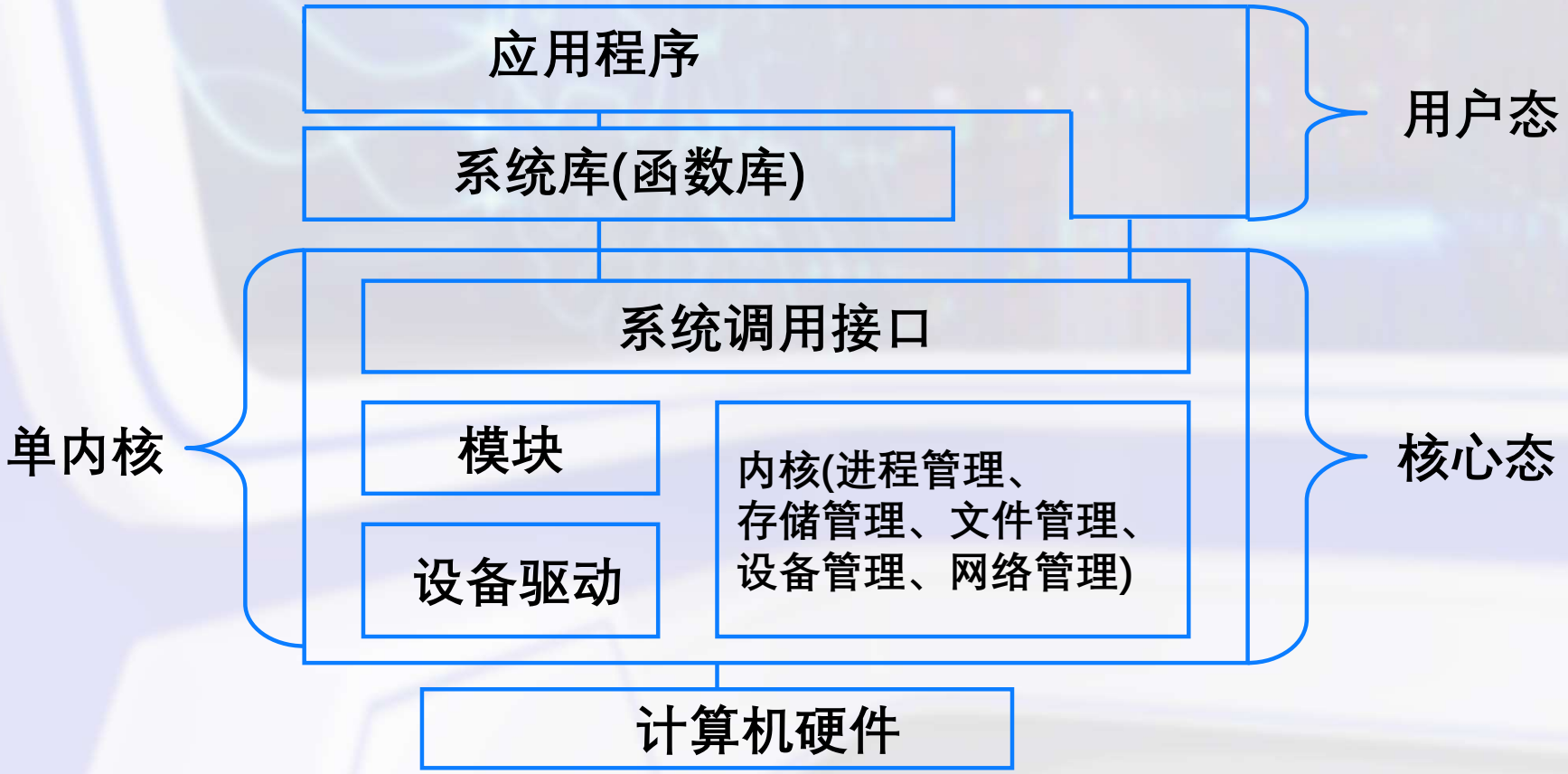
- ✓ 把整体问题局部化
- ✓ 层次结构和单向依赖性，使模块间的依赖和调用关系更为清晰规范

层次式结构的操作系统

Linux操作系统的结构



层次式结构的操作系统



- Linux整体是层次式结构
- 但是Linux内核内部是一个整体式结构
- 内核中多个模块可以彼此调用

层次式结构的操作系统

内核的组织方式是整体式结构

- Linux内核由模块组成
- 每个模块可以单独编译
- 模块用链接程序连在一起成为目标程序

内核是基于过程的开放结构

- ✓ 有利于不同的人参与不同过程的开发
- ✓ 允许任何人对其进行修改和完善

操作系统的结构设计

本讲内容

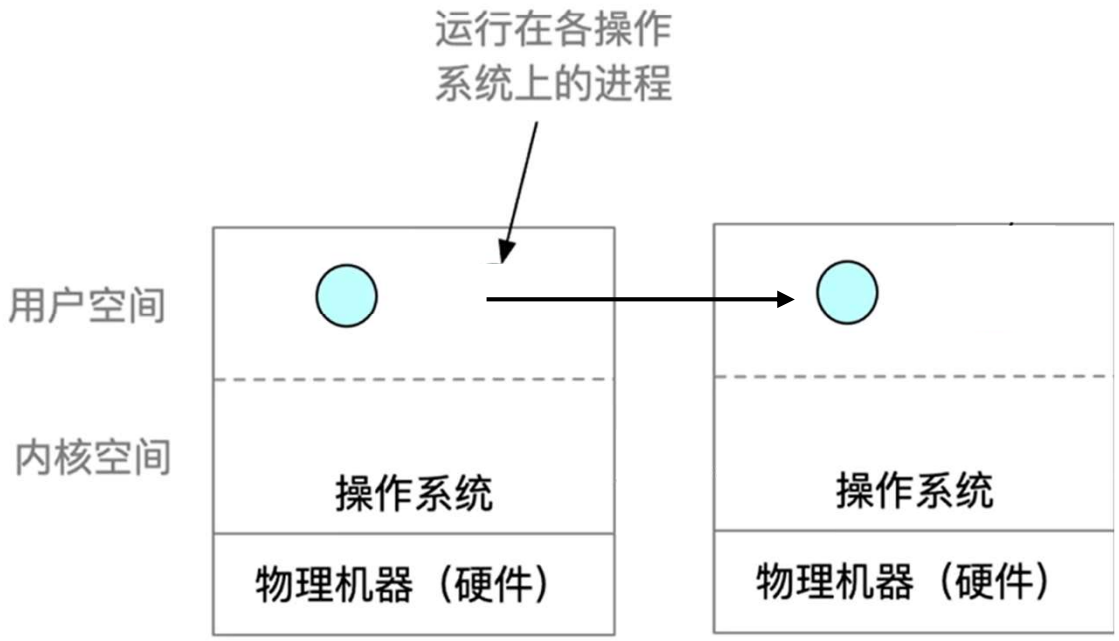
1. 整体式结构
2. 层次式结构
3. 虚拟机结构
4. 客户/服务器结构
5. 微内核结构

虚拟机结构的操作系统

定义

- 虚拟化是计算机资源的抽象方法
- 虚拟机是通过软件模拟的、运行在隔离环境中的计算机系统
- 实体计算机中能够完成的工作在虚拟机中都能够实现

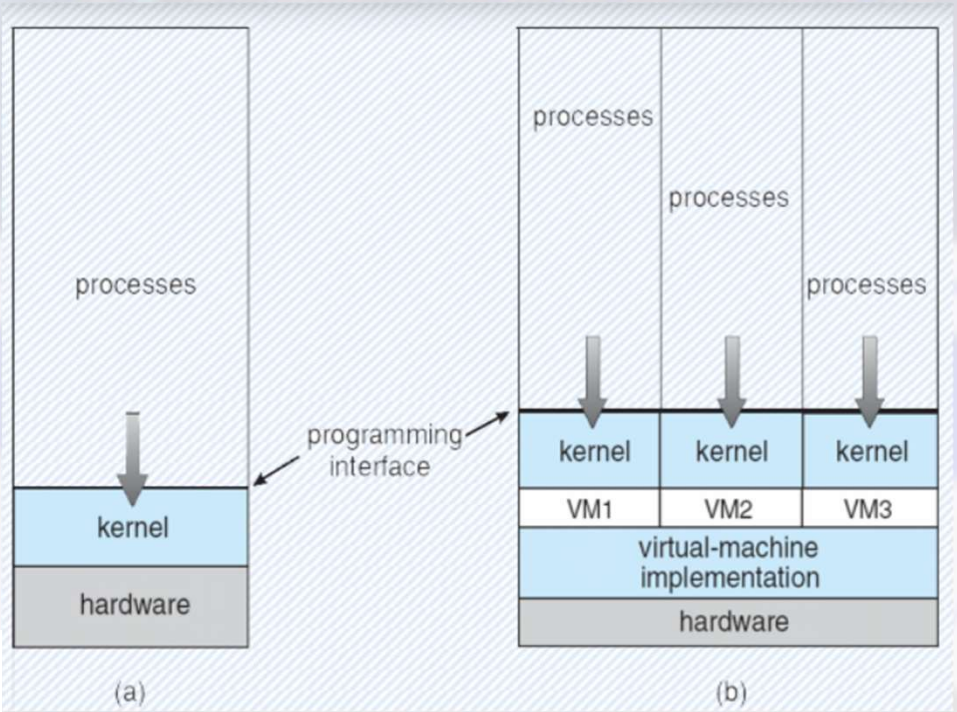
传统计算机



虚拟机结构的操作系统

方法

- 在裸机上层层扩展软件
- 可采用层次化结构的设计方法来实现
- 经过虚拟化后的逻辑资源对用户隐藏了不必要的硬件细节、物理上的差异

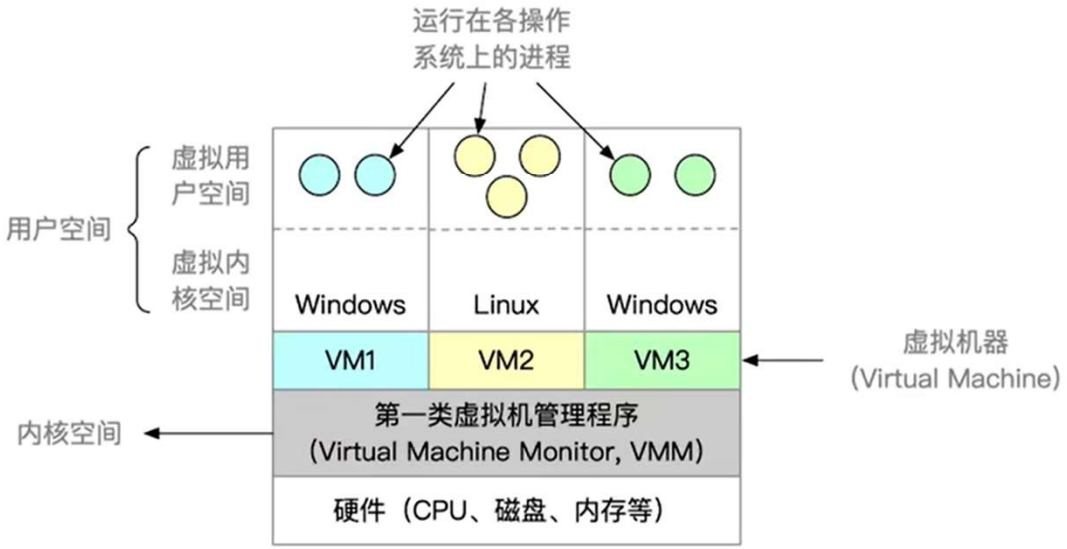


- 在同一个平台上构建出多个虚拟机，最终每个用户用到某个虚拟机
- 让程序员在虚拟机中进行更快的开发
- Unix Linux系统中都应用到这样一种虚拟机的思想

虚拟机

虚拟机：使用虚拟化技术，将一台物理机器虚拟化为多台虚拟机器（Virtual Machine, VM），每个虚拟机器都可以独立运行一个操作系统

同义术语：虚拟机管理程序/虚拟机监控程序/Virtual Machine Monitor/Hypervisor

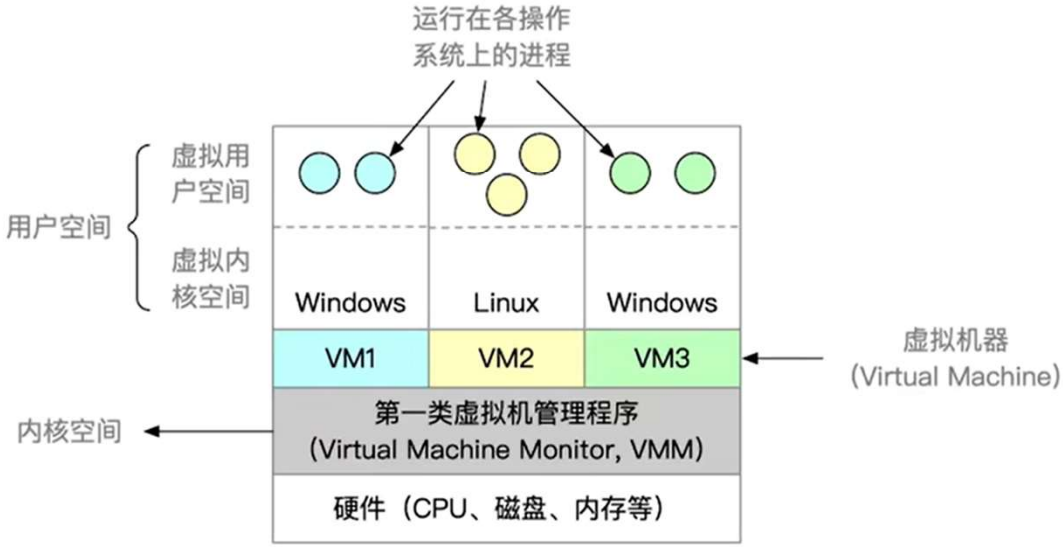


第一类VMM，直接运行在硬件上

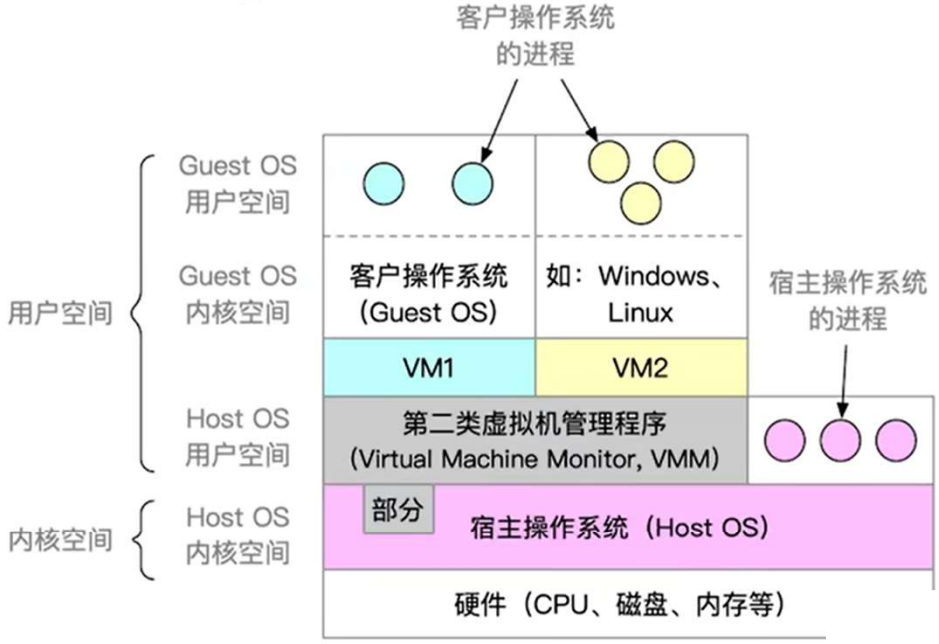
虚拟机

虚拟机：使用虚拟化技术，将一台物理机器虚拟化为多台虚拟机器（Virtual Machine, VM），每个虚拟机器都可以独立运行一个操作系统

同义术语：虚拟机管理程序/虚拟机监控程序/Virtual Machine Monitor/Hypervisor



第一类VMM，直接运行在硬件上



第二类VMM，运行在宿主操作系统上

常用的虚拟机软件

学生们常用的第二类VMM：
VirtualBox、VMware

- VM ware Player（官方提供的免费个人版）虚拟机下载以及配置虚拟机全流程（以linux为例）_vmware个人免费版-CSDN博客
- ubuntu 20.04.3 安装教程(本人一步一步安装记录...)_installation type-CSDN博客
- 南京邮电大学开源软件镜像站 | Njupt Open Source Mirror

操作系统的结构设计

本讲内容

1. 整体式结构
2. 层次式结构
3. 虚拟机结构
4. 客户/服务器结构
5. 微内核结构

客户/服务器结构的操作系统

基本思想

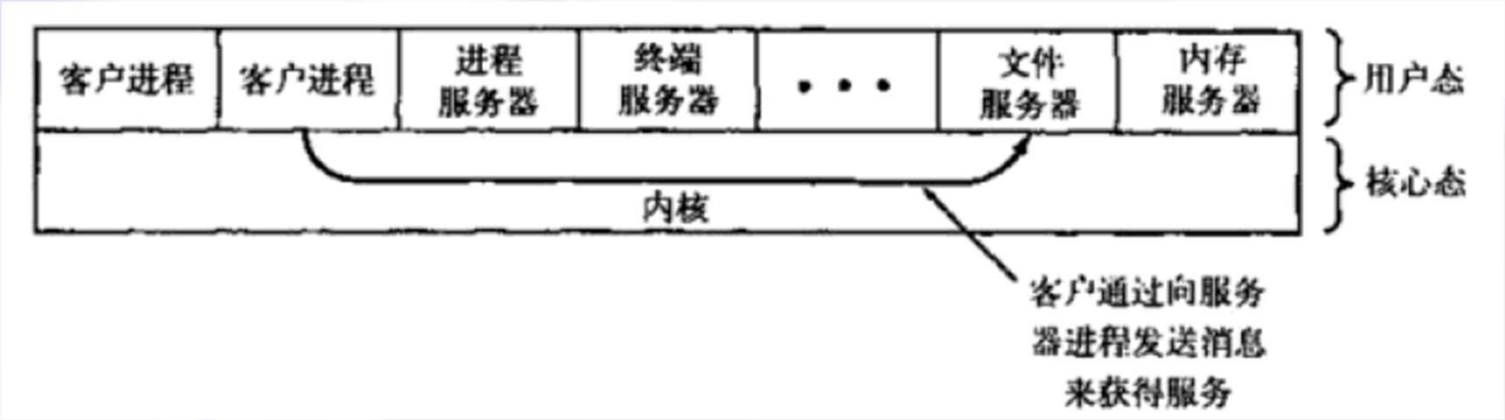
操作系统分成两大部分

- 运行在**用户态**并以客户/服务器方式活动的进程
  - 用户态中，存在客户进程(被服务)和 服务进程(服务)
  - 借鉴网络结构思想
- 运行在**核心态**的内核

客户/服务器结构的操作系统

- 除内核外，操作系统的其它部分被分成若干相对独立的**进程**，每一个进程实现一类服务，称**服务器进程**。**用户进程**也在该层并以客户/服务器方式活动。

- 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，在满足客户的要求后再通过内核发送消息把结果返回给用户。

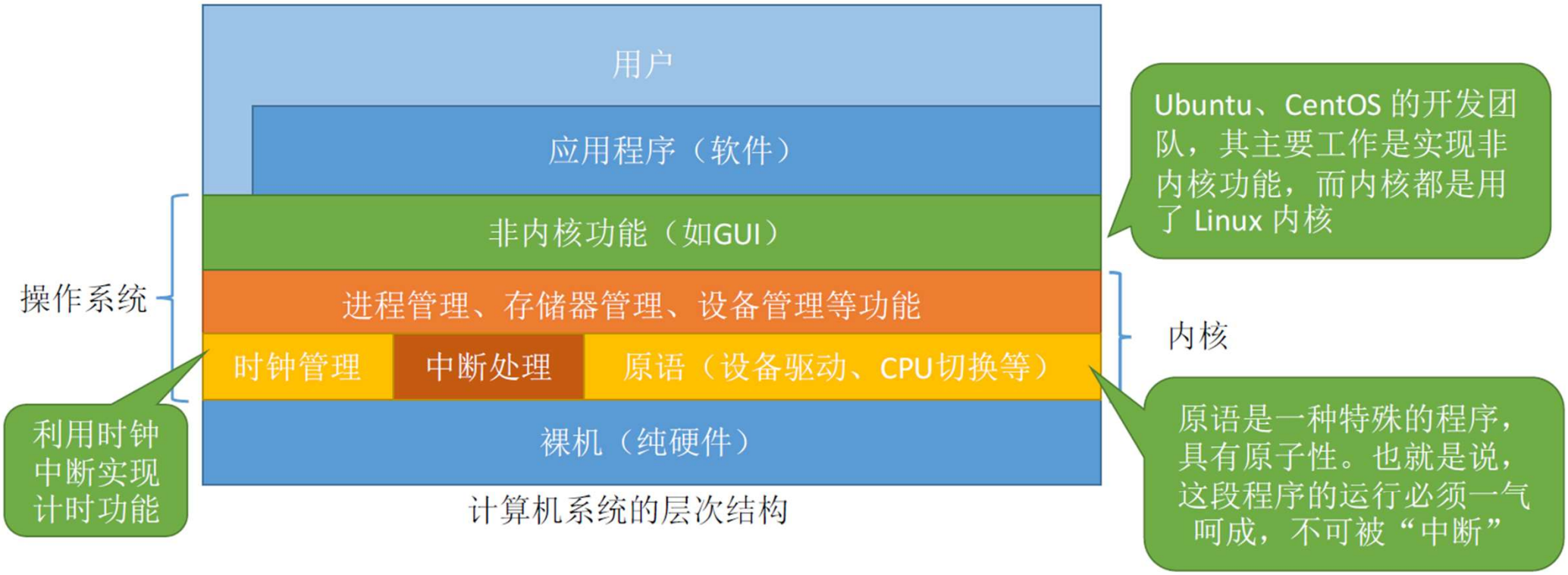


操作系统的结构设计

本讲内容

1. 整体式结构
2. 层次式结构
3. 虚拟机结构
4. 客户/服务器结构
5. 微内核结构

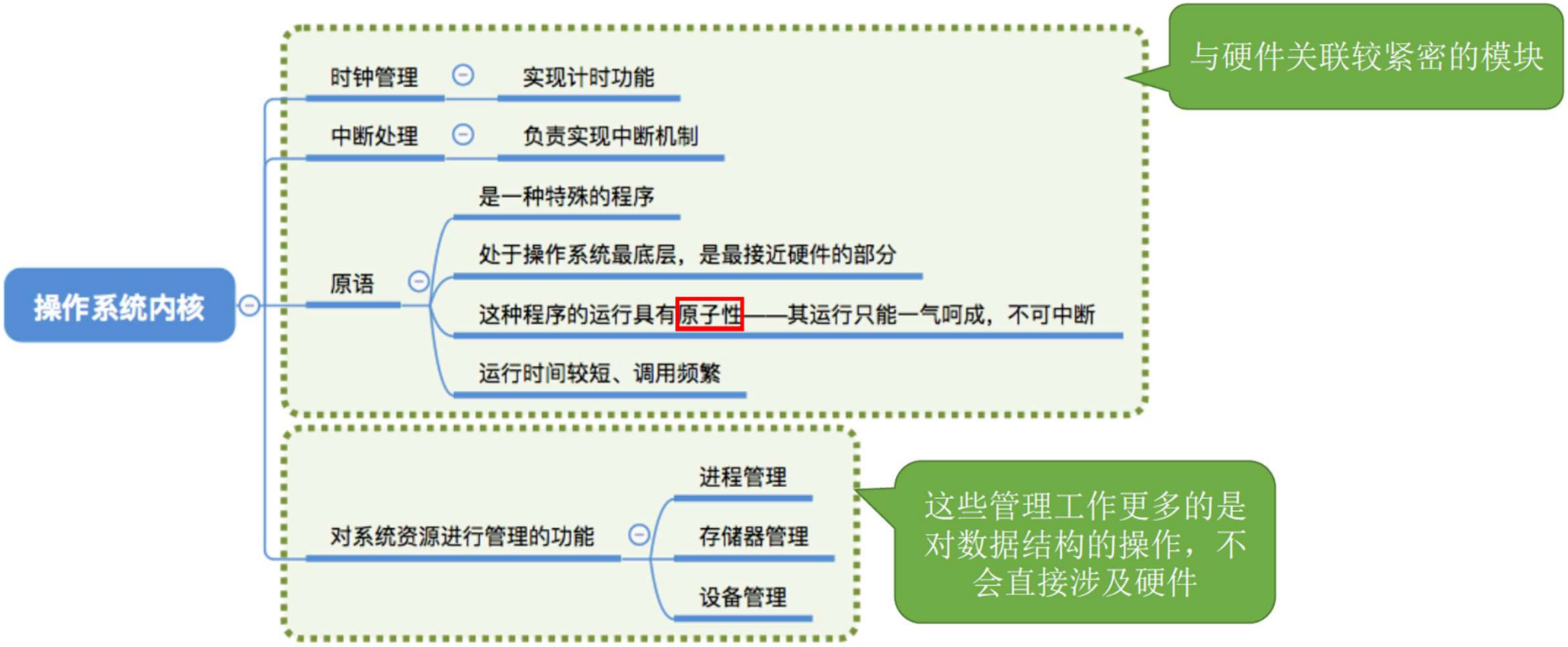
操作系统的内核



计算机系统的层次结构

操作系统的内核

内核是操作系统最基本、最核心的部分。
实现操作系统内核功能的那些程序就是内核程序。



操作系统的内核

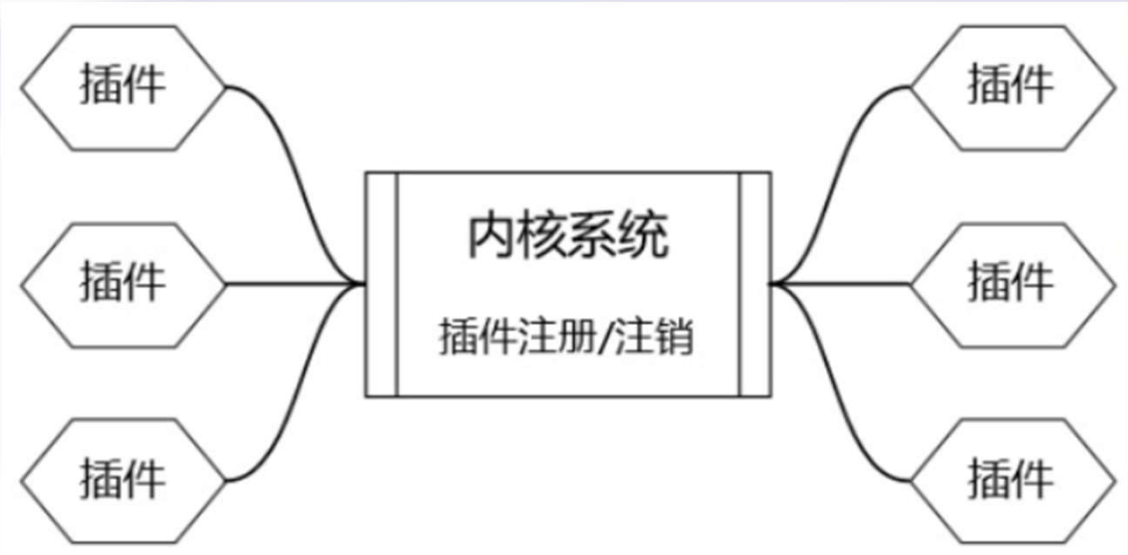
计算机系统的层次结构

注意：
操作系统内核需要运行在内核态
操作系统的非内核功能运行在用户态

微内核结构的操作系统

定义

- 把操作系统中的内存管理、设备管理、文件系统等功能模块尽可能地从内核中分离出来
- 在内核只保留少量最基本的功能，使内核变得简洁可靠



微内核结构的操作系统

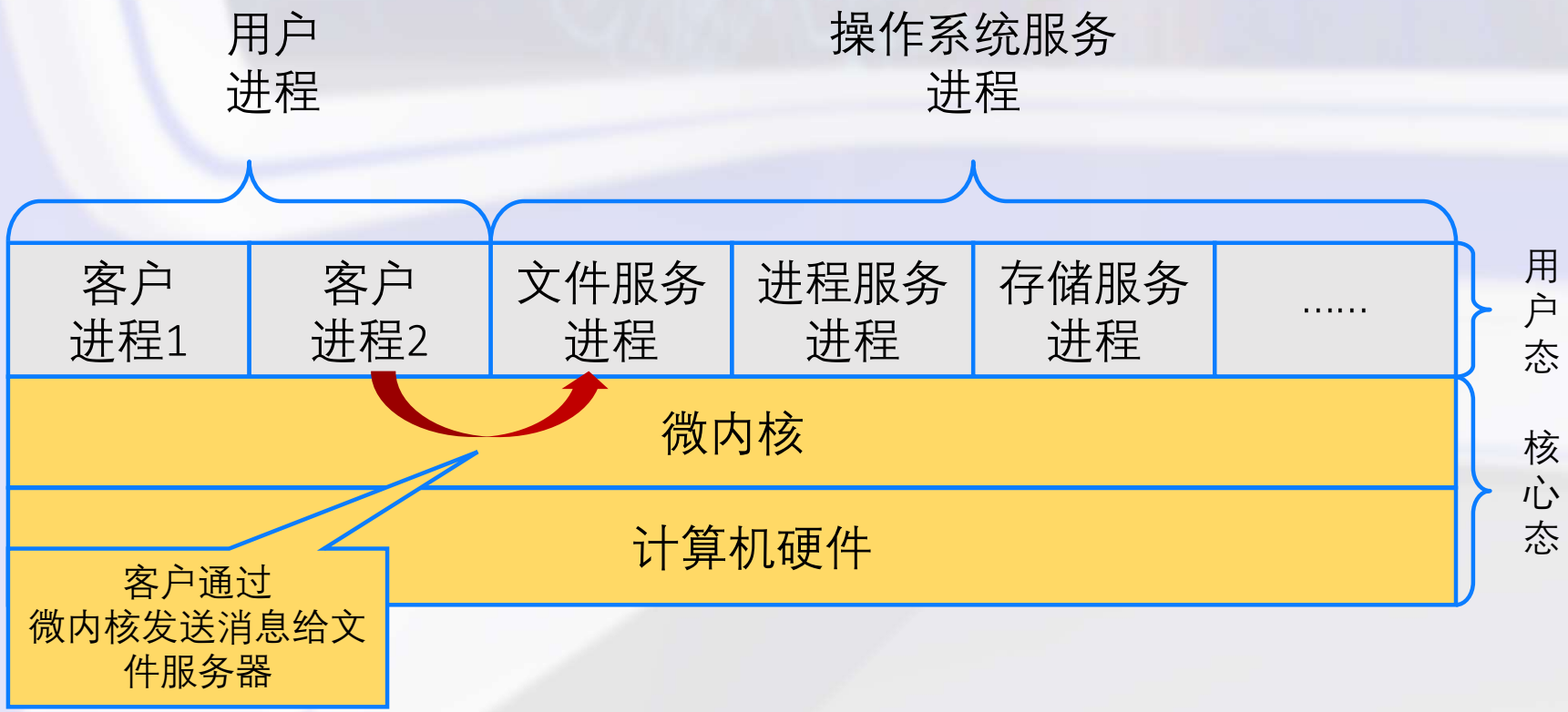
优点

- 充分的模块化，可独立更换任一模块而不会影响其它模块，从而方便第三方设计、开发各个模块
- 未被使用的模块功能不必运行，大幅度减少系统的内存需求
- 增强可移植性，移植时主要对微内核部分进行修改即可，减轻移植工作量。

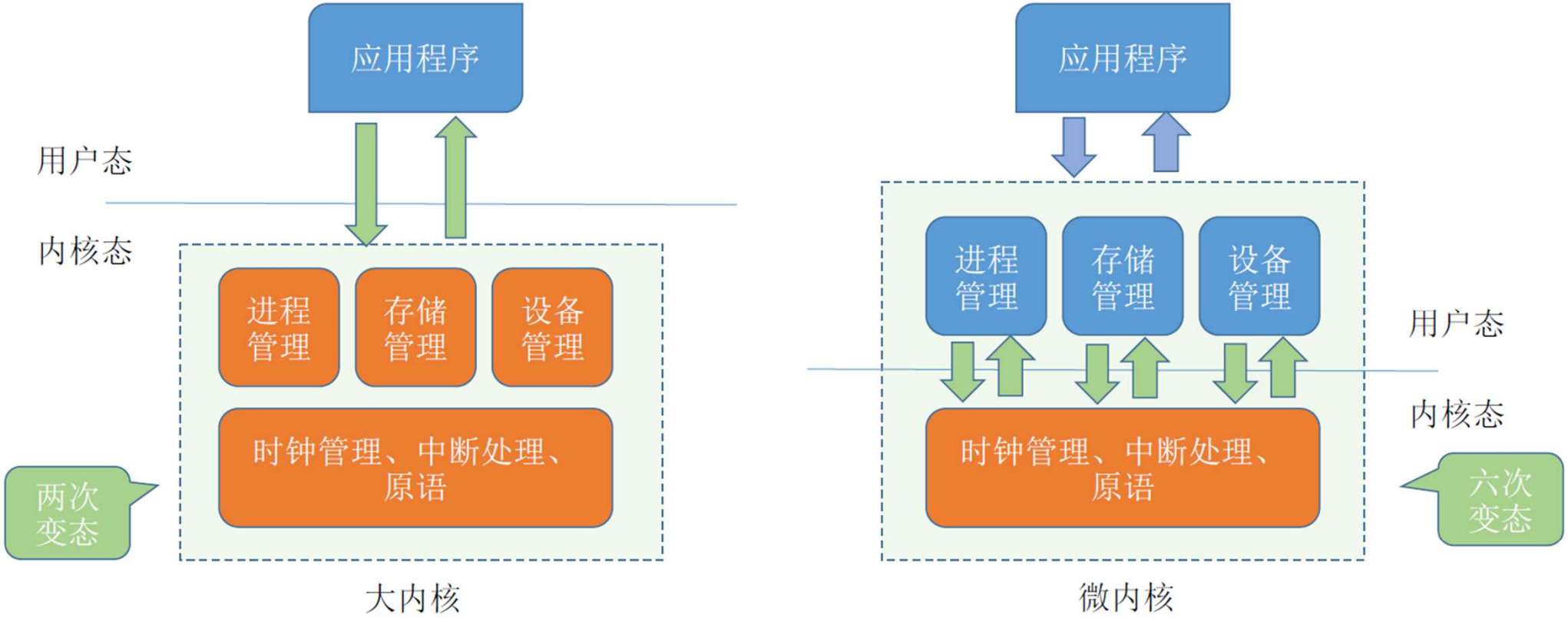
微内核结构的操作系统

客户/服务器及微内核结构

- 用户进程通过微内核把请求发给服务器进程，去索取服务
- 服务进程通过微内核把结果转发回给用户
- Windows 客户服务器结构 和 用到了微内核思想



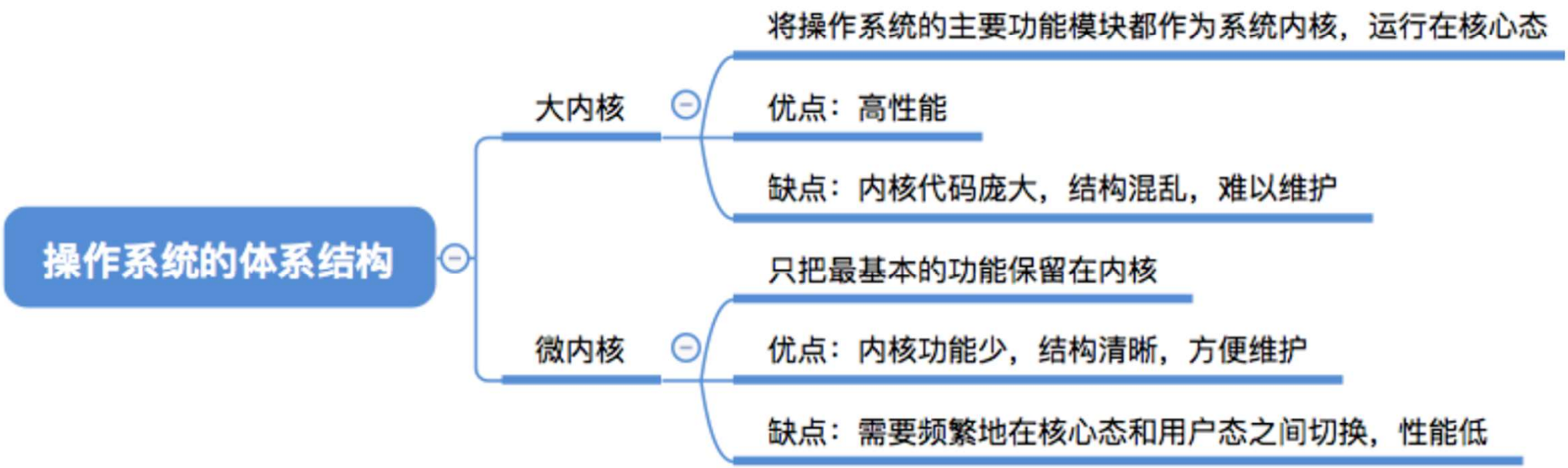
操作系统的体系结构



一个故事：现在，应用程序想要请求操作系统的服务，这个服务的处理同时涉及到进程管理、存储管理、设备管理

注意：变态的过程是有成本的，要消耗不少时间，频繁地变态会降低系统性能

知识回顾与重要考点



典型的大内核/宏内核/单内核 操作系统： Linux、UNIX
典型的 微内核 操作系统： Windows NT（企业级的网络操作系统）
鸿蒙OS是一款基于微内核的全场景分布式操作系统