

Chapter 1 计算机系统概述

Chapter 1 计算机系统概述

一、操作系统基本概念

1. 操作系统的概念
2. 功能和目标
 - (1) 作为计算机系统资源的管理者
 - (2) 作为用户与计算机硬件系统之间的接口
 - (3) 实现对计算机资源的扩充
3. 操作系统的特征
 - (1) 并发 (Concurrence)
 - (2) 共享 (Sharing)
 - (3) 虚拟 (Virtual)
 - (4) 异步 (Asynchronism)

二、操作系统的发展历程

三、操作系统的运行环境

1. 处理器运行模式
2. 中断和异常的概念
3. 系统调用

四、操作系统的体系结构

1. 分层法
2. 模块化
3. 宏内核
4. 微内核
 - (1) 微内核的基本概念
 - (2) 微内核的基本功能
 - (3) 微内核操作系统的优点

5. 外核

五、操作系统引导

六、虚拟机

1. 第一类虚拟机管理程序
2. 第二类虚拟机管理程序

七、本章难点

一、操作系统基本概念

1. 操作系统的概念

计算机系统自下而上可以分为4部分：硬件，操作系统，应用程序，用户

操作系统：是指控制和管理整个计算机系统的硬件与软件资源，合理的组织、调度，计算机的工作与资源的分配，进而为用户和其他软件提供方便的接口与环境的程序集合。

是计算机系统中最基本的系统软件

2. 功能和目标

(1) 作为计算机系统资源的管理者

- 处理机管理
 - 多道程序环境下，处理机的分配和运行都以进程（线程）为基本单位，对处理机的管理可以归纳为对进程的管理
 - 并发是指在计算机内同时运行多个程序，进程何时创建，如何撤销，怎么管理，避免冲突，合理共享，就是进程管理最重要的任务
 - **包括：进程控制，进程同步，进程通信，死锁处理，处理机调度**
 - 存储器管理
 - 存储器管理是为了给对程序的运行提供良好环境，提高内存利用率
 - **包括：内存分配与回收，地址映射，内存保护与共享，内存扩充**
 - 文件管理
 - 计算机中的信息都以文件形式存在
 - **包括：文件存储空间的管理，目录管理，文件读写管理，保护**
 - 设备管理
 - 主要任务：完成用户的I/O请求，方便用户使用各种设备，提供设备利用率
 - **包括：缓冲管理，设备分配，设备处理，虚拟设备**
-

(2) 作为用户与计算机硬件系统之间的接口

操作系统提供用户接口，分为两类：命令接口可以让用户使用来组织和控制作业的执行，程序接口可以让编程人员请求操作系统服务。

- 命令接口
 - 使用命令进行作业控制的方式有两种：
 - 联机控制方式：（交互式命令接口），适用于分时或实时系统的接口，由一组键盘操作命令组成（雇主说一句话，工人做一件事，做出反馈，强调交互）
 - 脱机命令接口：（批处理命令接口），适用于批处理系统，由一组作业控制命令组成（雇主将工人要做的事情写在清单上，工人逐条完成这些事情）
- 程序接口
 - 由一组系统调用（也称为广义指令）组成，用户通过在程序中使用这些系统调用来请求OS为其提供服务。
 - 最流行的是图形化界面（GUI），即图形接口，图形接口不是操作系统的一部分，但是图形接口所调用的系统调用命令是OS的一部分

(3) 实现对计算机资源的扩充

没有任何软件支持的计算机称为裸机，将覆盖了软件的机器称为扩充机器或虚拟机。

3. 操作系统的特征

(1) 并发 (Concurrence)

指两个或多个事件在同一时间间隔内发生，多道程序环境下，利用其I/O操作而暂停执行时的CPU空档时间，再调度另一个程序运行，使得多道程序交替运行，CPU保持忙碌状态。

区分并发与并行：

实现进程的并行，需要有硬件的支持，如多流水线，多处理机环境

支持多道程序的单处理机环境下，一段时间内，宏观上有多道程序在同时执行，而在每个时刻，实际仅能有一道程序执行，因此微观上这些程序仍是分时交替执行的。

(2) 共享 (Sharing)

是指系统中的资源可供内存中的多个并发执行的进程共同使用，分为两种方式：

- 互斥共享方式
 - 一段时间内只允许一个进程访问该资源，资源称为临界资源
- 同时访问方式
 - 这类资源允许一段时间内由多个进程"同时"访问。
 - 这里所说的"同时"通常是宏观上的,而在微观上,这些进程可能是交替地对该资源进行访问,即"分时共享"的。
 - 可供多个进程"同时"访问的典型资源是磁盘设备,一些用重入代码编写的文件也可被"同时"共享,即允许若干用户同时访问该文件。

(3) 虚拟 (Virtual)

将一个物理上的实体变为若干逻辑上的对照物。操作系统的虚拟技术可以归纳为：

- 时分复用技术：如虚拟处理器，
- 空分复用技术：如虚拟存储器

(4) 异步 (Asynchronism)

多道程序环境允许多个程序并发执行，但由于资源有限，进程的执行不是一贯到底的，而是走走停停，以不可预知的速度向前推进，这就是进程的异步性

并发和共享是操作系统两个最基本的特征,两者之间互为存在的条件:

1资源共享是以程序的并发为条件的,若系统不允许程序并发执行,则自然不存在资源共享问题:

2若系统不能对资源共享实施有效的管理,则必将影响到程序的并发执行,甚至根本无法并发执行。

二、操作系统的发展历程

三、操作系统的运行环境

1. 处理器运行模式

CPU执行两种不同性质的程序：操作系统内核程序，应用程序

- 特权指令：不允许用户直接使用的指令，如I/O指令，关中断，内存清零，存取用于内存保护的寄存器，修改程序状态字寄存器等的指令
- 非特权指令：允许用户直接使用，不能直接访问系统中的软硬件资源，仅限于用户的地址空间

具体实现上，将CPU的运行模式划分为用户态（目态）和内核态（管态，核心态），操作系统是分层结构。

- 与硬件联系紧密的：时钟管理，中断处理，设备驱动，处于低层
- 运行频率较高的程序：进程管理，存储器管理，设备管理，位于高层。

这两部分构成操作系统的内核，这部分指令运行在内核态。

操作系统的内核包括以下4部分内容：

- 时钟管理
 - 计时
 - 实现进程切换，如分时OS中，时间片轮转调度，实时OS中按截止时间控制运行，批处理OS中，通过时钟管理来衡量一个作业的运行程度。
- 中断机制
 - 目的：提供CPU利用率，使得CPU在I/O操作执行时执行其他指令
 - 如：键盘或鼠标信息的输入，进程的管理与调度，系统功能的调用，设备驱动，文件访问
 - 中断机制中，只有一小部分功能属于内核，负责保护和回复中断现场的信息，转移控制权到相关的处理程序，减少中断的处理时间，提高并行处理能力。
- 原语
 - 按层次结构设计的操作系统，底层是一些可被调用的公用小程序，各自完成一个规定的操作，将这些程序称为原语。
 - 特点：
 - 处于OS最底层，最接近硬件
 - 运行具有原子性，其操作只能一气呵成
 - 运行时间短，调度频繁
 - 定义原语的直接方法是关中断，系统中的设备驱动，CPU切换，进程通信，等功能中的部分操作，都可以定义为原语
- 系统控制的数据结构与处理
 - 为实现有效管理进程中的数据结构，需要一些基本操作：
 - 进程管理
 - 存储器管理
 - 设备管理

内核态指令实际上包括系统调用类指令和一些针对时钟，中断和原语的操作指令。

2. 中断和异常的概念

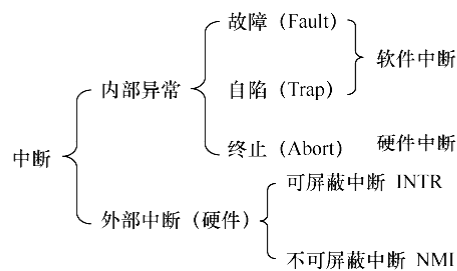


图 1.2 内中断和外中断的联系与区别

3. 系统调用

系统调用是操作系统提供给应用程序使用的接口。

凡是与共享资源有关的操作，都必须通过系统调用的方式向OS提出服务请求，由操作系统代为完成，并将处理结果返回给应用程序。

系统调用的功能：

- 设备管理：完成设备的请求和释放，以及设备启动等功能
- 文件管理：完成文件的读写，创建，删除
- 进程控制：完成进程的创建，撤销，阻塞，唤醒等功能
- 进程通信：完成进程之间的消息传递或信号传递等功能
- 内存管理：完成内存的分配，回收，以及获取作业占用内存区的大小和起始地址

系统调用运行在内核态

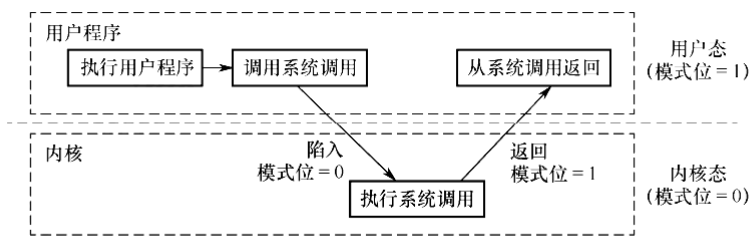


图 1.3 系统调用的执行过程

系统调用的处理过程：

- 第一步：
 - 用户将系统调用号和所需参数压入堆栈，
 - 调用实际的调用实际的调用指令，执行第一个陷入指令，将CPU转为内核态
 - 由硬件和内核程序保护被中断进程的现场，将PC,PSW（程序状态字）及通用寄存器压入堆栈
- 第二步：
 - 分析调用类型，转入相应系统调用处理子程序
 - （系统入口中有一张系统调用入口表，根据系统调用号可以找到系统调用中处理子程序的入口地址）
- 第三步：
 - 在系统调用子程序执行完后，回复被中断的或设置新进程的CPU现场
 - 返回中断进程或新进程，继续执行

由用户态转向内核态的例子：

- 用户要求操作系统的服务，即系统调用
- 发生一次中断
- 用户程序中产生了一个错误状态
- 用户程序中企求执行一条特权指令

四、操作系统的体系结构

1. 分层法

将操作系统分为若干层，底层（0）为硬件，顶层（层N）为用户接口，每层只能调用紧邻他的低层的功能和服务（单向依赖）。

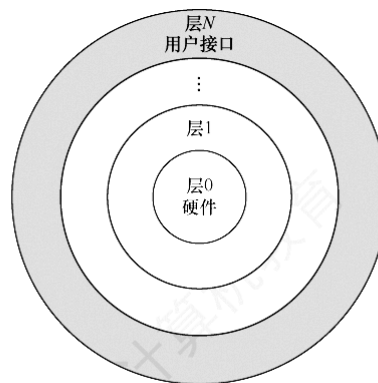


图 1.4 分层的操作系统

优点：

- 便于系统的调试和验证，简化系统的设计和实现
- 易扩充，易维护，在系统重增加，修改，替换一层中的模块或整层时，只要不改变相应层间的接口，就不会影响其它层。

问题：

- 合理定义各层比较困难
- 效率较差，执行一个功能需要穿越多层，每层之间都有通信机制，增加开销，降低效率

2. 模块化

将操作系统按功能划分为若干具有一定独立性的模块。

各模块具有某方面的管理功能，并规定好各模块间的接口，使各模块可以通过接口通信并且可以划分为子模块。

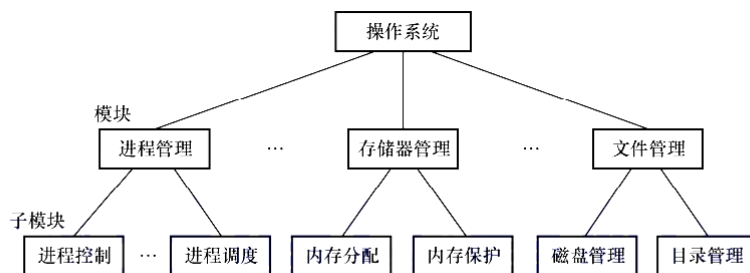


图 1.5 由模块、子模块等组成的模块化操作系统结构

模块划分需要充分考虑模块的独立性，越独立，交互就越少，系统结构就越清晰

衡量模块独立的标准：

- 内聚性：模块内部各部分联系的紧密程度，内聚性越高，模块独立性越好
- 耦合度：模块间相互联系和相互影响的程度，耦合度越低，模块独立性越好

优点：

- 提高了操作系统设计的正确性，可理解性，可维护性
- 增强了操作系统的可适应性
- 加速操作系统的开发过程

问题：

- 模块间的接口规定很难满足对接口的实际需求
- 各模块设计者齐头并进，每个决定无法建立在上一个已验证的正确决定的基础上，无法找到一个可靠的决定顺序

3. 宏内核

从操作系统的内核架构来划分：宏内核，微内核

- 宏内核：
 - 单内核，大内核，将操作系统的主要功能模块都作为一个紧密联系的整体运行在内核态。

各管理模块之间共享信息，能够有效利用相互之间的有效特性，具有无可比拟的新能特性

操作系统的设计规模急剧增长，将一些非核心功能移到用户空间。

主流操作系统Window，Android，iOS，macOS，Linux都是基于宏内核的架构。

宏内核架构如今遇到越来越多的挑战和困难，微内核优势似乎越来越明显。

谷歌的Fuchsia和华为的鸿蒙OS，都瞄准微内核架构。

4. 微内核

(1) 微内核的基本概念

- 是指将内核中最基本的功能保留在内核，而将那些不需要在内核态执行的功能移到用户态执行，从而降低内核的设计复杂性。
- 那些移出内核的操作系统代码根据分层的原则被划分成若干服务程序，它们的执行相互独立，交互则都借助于微内核进行通信。

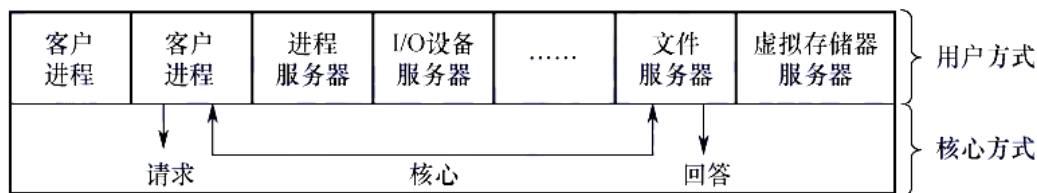


图 1.6 单机环境下的客户/服务器模式

微内核将操作系统划分为两部分：微内核和多个服务器。

- 微内核：
 - 指精心设计的，能实现OS最基本核心功能的小型内核
 - 包括：
 - 与硬件处理紧密相关的部分
 - 一些较基本的功能
 - 客户和服务端之间的通信
- 操作系统中的绝大部分功能都放在微内核外的一组服务器(进程)中实现
- 如用于提供对进程(线程)进行管理的进程(线程)服务器、提供虚拟存储器管理功能的虚拟存储器服务器等，它们都是作为进程来实现的，运行在用户态
- 客户与服务端之间 是借助微内核提供的消息传递机制来实现交互的。

在微内核结构中，为了实现高可靠性，只有微内核运行在内核态，其余模块都运行在用户态，一个模块中的错误只会使这个模块崩溃，而不会使整个系统崩溃。

(2) 微内核的基本功能

- 进程（线程）管理：
 - 通信，切换，调度，以及多处理机之间的同步，放入微内核，而对于用户进程如何分类，优先级确认方式，是策略问题，放入微内核外的进程管理服务器中。
- 低级存储器管理：
 - 微内核中只配置最低级的存储器管理机制，，如用于实现将逻辑地址变换为物理地址等的页表机制和地址变换机制，依赖于硬件的，放入微内核。
 - 实现虚拟存储器管理的策略，采用何种置换算法，何种内存分配与策略，放入微内核外的存储器管理服务器中。
- 中断和陷入处理：
 - 微内核 OS 将与硬件紧密相关的一小部分放入微内核，此时微内核的主要功能是捕获所发生的中断和陷入事件，并进行中断响应处理
 - 识别中断或陷入的事件后，再 发送给相关的服务器来处理，所以中断和陷入处理也应放入微内核。

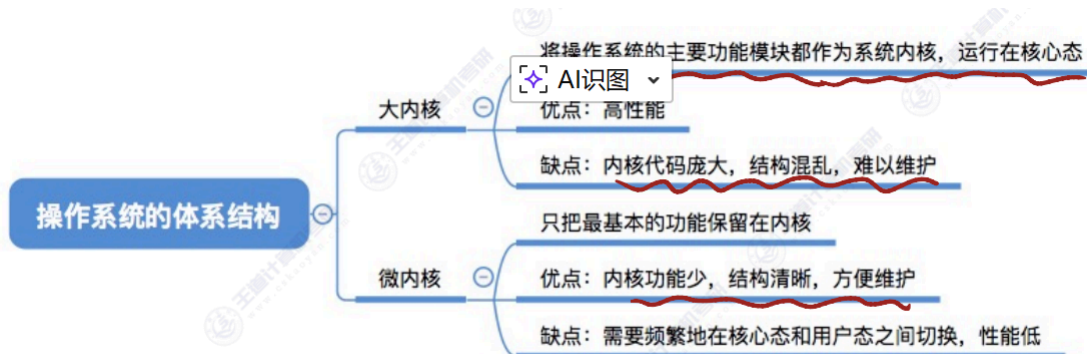
(3) 微内核操作系统的优点

- 扩展性和灵活性。
 - 许多功能从内核中分离出来，当要修改某些功能或增加新功能时，只需在相应的服务器中修改或新增功能，或再增加一个专用的服务器，而无须改动内核代码。
- 可靠性与安全性

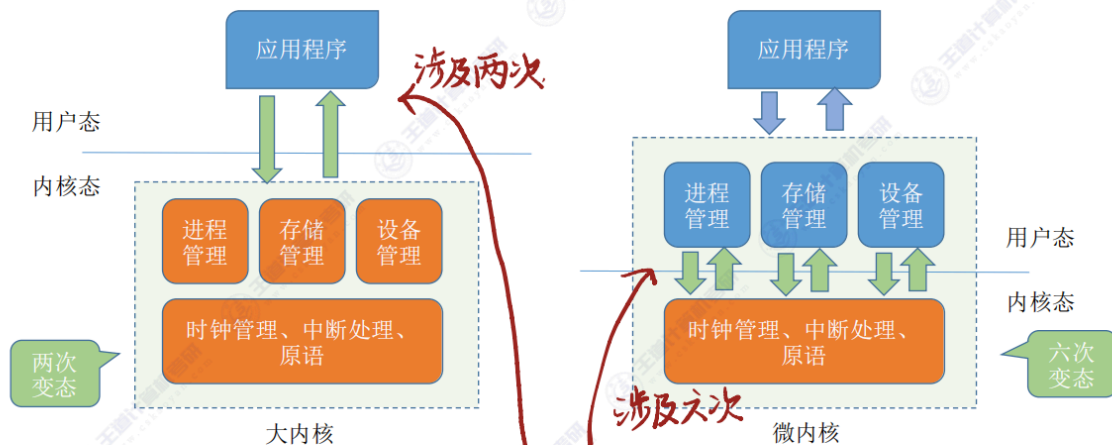
- 可移植性
 - 与CPU 和I/O 硬件有关的代码均放在内核中，而其他各种服务器均与硬件平台 无关，因而将操作系统移植到另一个平台上所需做的修改是比较小的。
- 分布式计算
 - 客户和服务之间、服务器和服务器之间的通信采用消息传递机制，这就 使得微内核系统能很好地支持分布式系统和网络系统。

微内核结构的主要问题是性能问题，因为需要频繁地在内核态和用户态之间进行切换，操 作系统的执行开销偏大。

为了改善运行效率，可以将那些频繁使用的系统服务移回内核，从而 保证系统性能，但这又会使微内核的容量明显地增大。



大内核 vs 微内核 变态=CPU状态转换



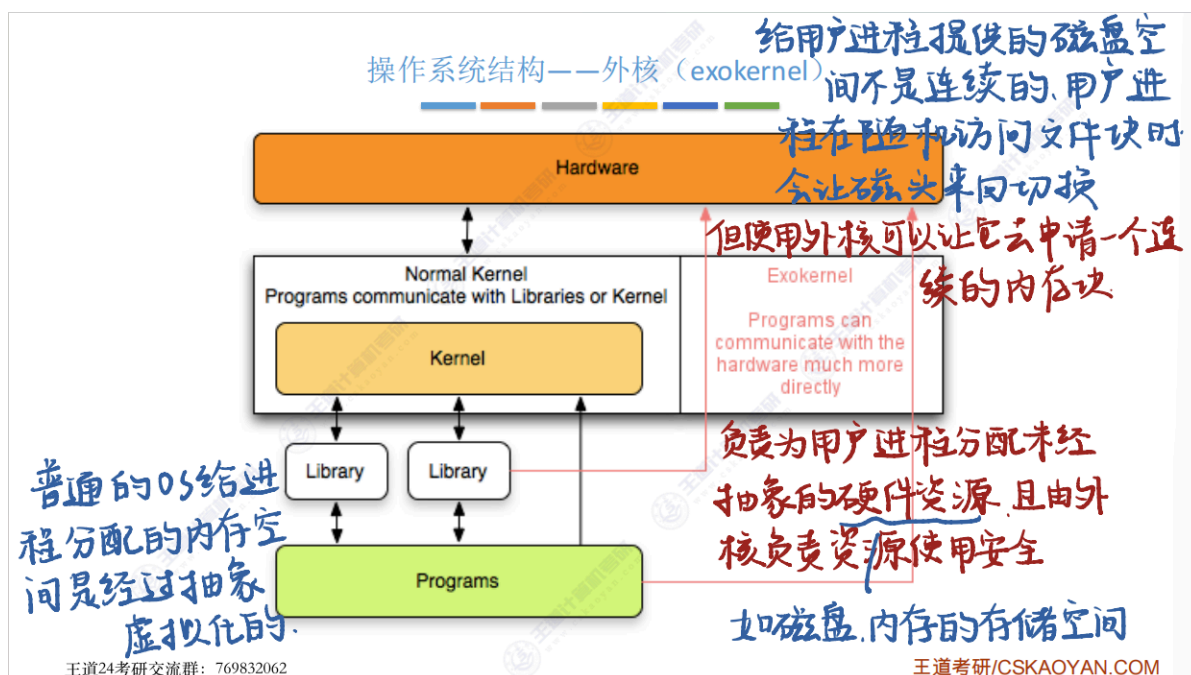
一个故事：现在，应用程序想要请求操作系统的服务，这个服务的处理同时涉及到进程管理、存储管理、设备管理

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

5. 外核

在底层，一种称为外核 (exokernel) 的程序在内核态中运行。它的任务是为虚拟机分配 资源，并检查这些资源使用的安全性，以确保没有机器会使用他人的资源。

每个用户的虚拟机 可以运行自己的操作系统，但限制只能使用已经申请并且获得分配的那部分资源。



其他设计中, 每个虚拟机系统都认为它拥有完整的磁盘或其他资源, 这样的虚拟机监控程序就必须维护一张表格来重映像磁盘地址。

有了外核, 这个重映射机制就不需要了。外核只需要记录分配给各个虚拟机的有关资源即可

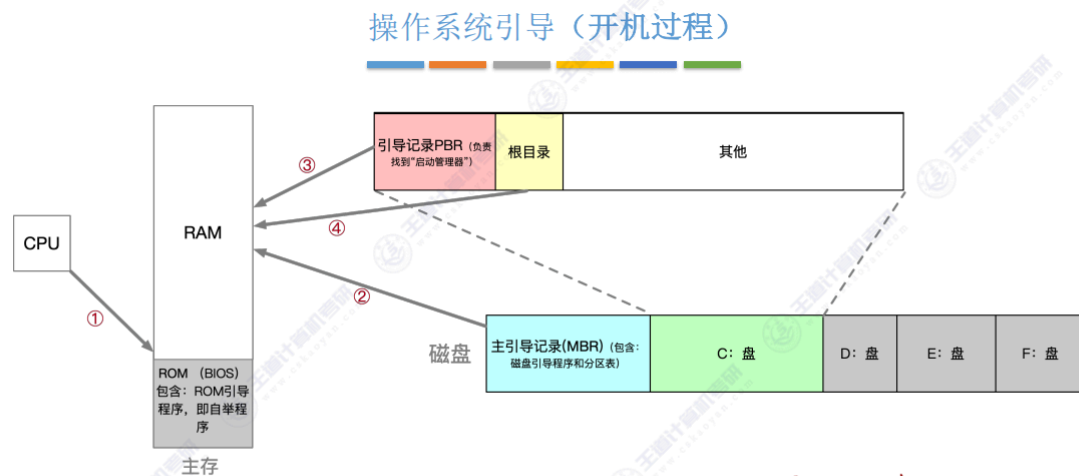
操作系统结构			
	特性、思想	优点	缺点
分层结构	内核分多层, 每层可单向调用更低一层提供的接口	<ul style="list-style-type: none"> 1. 便于调试和验证, 自底向上逐层调试验证 2. 易扩充和易维护, 各层之间调用接口清晰固定 	<ul style="list-style-type: none"> 1. 仅可调用相邻低层, 难以合理定义各层的边界 2. 效率低, 不可跨层调用, 系统调用执行时间长
模块化	将内核划分为多个模块, 各模块之间相互协作。 内核 = 主模块 + 可加载内核模块 主模块: 只负责核心功能, 如进程调度、内存管理 可加载内核模块: 可以动态加载新模块到内核, 而无需重新编译整个内核	<ul style="list-style-type: none"> 1. 模块间逻辑清晰易于维护, 确定模块间接口后即可多模块同时开发 2. 支持动态加载新的内核模块 (如: 安装设备驱动程序、安装新的文件系统模块到内核), 增强OS适应性 3. 任何模块都可以直接调用其他模块, 无需采用消息传递进行通信, 效率高 	<ul style="list-style-type: none"> 1. 模块间的接口定义未必合理、实用 2. 模块间相互依赖, 更难调试和验证
宏内核 (大内核)	所有的系统功能都放在内核里 (大内核结构的OS通常也采用了“模块化”的设计思想)	<ul style="list-style-type: none"> 1. 性能高, 内核内部各种功能都可以直接相互调用 	<ul style="list-style-type: none"> 1. 内核庞大功能复杂, 难以维护 2. 大内核中某个功能模块出错, 就可能致整个系统崩溃
微内核	只把中断、原语、进程通信等最核心的功能放入内核。进程管理、文件管理、设备管理等功能以用户进程的形式运行在用户态	<ul style="list-style-type: none"> 1. 内核小功能少、易于维护, 内核可靠性高 2. 内核外的某个功能模块出错不会导致整个系统崩溃 	<ul style="list-style-type: none"> 1. 性能低, 需要频繁的切换用户态/核心态。用户态下的各功能模块不可以直接相互调用, 只能通过内核的“消息传递”来间接通信 2. 用户态下的各功能模块不可以直接相互调用, 只能通过内核的“消息传递”来间接通信
外核 (exokernel)	内核负责进程调度、进程通信等功能, 外核负责为用户进程分配未经抽象的硬件资源, 且由外核负责保证资源使用安全	<ul style="list-style-type: none"> 1. 外核可直接给用户进程分配“不虚拟、不抽象”的硬件资源, 使用户进程可以更灵活的使用硬件资源 2. 减少了虚拟硬件资源的“映射层”, 提升效率 	<ul style="list-style-type: none"> 1. 降低了系统的一致性 2. 使系统变得更复杂

五、操作系统引导

操作系统引导是指计算机利用 CPU 运行特定程序, 通过程序识别硬盘, 识别硬盘分区, 识别硬盘分区上的操作系统, 最后通过程序启动操作系统, 一环扣一环地完成上述过程。

- 激活CPU
 - 激活的CPU 读取 ROM 中的boot 程序, 将指令寄存器置为 BIOS (基本输入/ 输出系统)的第一条指令, 即开始执行BIOS 的指令。
- 硬件自测:
 - 激活的CPU 读取 ROM 中的boot 程序, 将指令寄存器置为 BIOS (基本输入/ 输出系统)的第一条指令, 即开始执行BIOS 的指令。
 - 然后进行通电自检, 检查硬件是否出现故障。

- 如有故障，主板会发出不同含义的蜂鸣，启动中止；
 - 如无故障，屏幕会显示CPU、内存、硬盘等信息。
- 加载带有操作系统的硬盘。
 - 通电自检后，BIOS 开始读取Boot Sequence(通过CMOS 里保存的启动顺序，或者通过与用户交互的方式)，将控制权交给启动顺序排在第一位的存储设备，
 - 然后 CPU 将该存储设备引导扇区的内容加载到内存中。
- 加载主引导记录 (MBR)
 - 硬盘以特定的标识符区分引导硬盘和非引导硬盘。
 - 主引导记录 MBR 的作用是告诉CPU 去硬盘的哪个主分区去找操作系统。
- 扫描硬盘分区表，并加载硬盘活动分区。
 - MBR 包含硬盘分区表，硬盘分区表以特定的标识符区分活动分区和非活动分区。
 - 主引导记录扫描硬盘分区表，进而识别含有操作系统的硬盘 分区(活动分区)。
 - 主引导记录扫描硬盘分区表，进而识别含有操作系统的硬盘 分区(活动分区)。
- 加载分区引导记录(PBR)
 - 读取活动分区的第一个扇区，这个扇区称为分区引导记录(PBR)，
 - 其作用是寻找并激活分区根目录下用于引导操作系统的程序(启动管理器)。
- 加载启动管理器。
 - 分区引导记录搜索活动分区中的启动管理器，加载启动管理器。
- 加载操作系统。
 - 将操作系统的初始化程序加载到内存中执行。



操作系统引导:

- ① CPU 从一个特定主存地址开始，取指令，执行ROM中的引导程序（先进行硬件自检，再开机）
- ② 将磁盘的第一块——主引导记录 读入内存，执行磁盘引导程序，扫描分区表
- ③ 从活动分区（又称主分区，即安装了操作系统的分区）读入分区引导记录，执行其中的程序
- ④ 从根目录下找到完整的操作系统初始化程序（即 启动管理器）并执行，完成“开机”的一系列动作

有没有插磁盘已内存条?

六、虚拟机

虚拟机的基本概念

虚拟机是指利用虚拟化技术，将一台物理机器虚拟化为多台虚拟机器，通过隐藏特定计算平台的实际物理特性，为用户提供抽象的、统一的、模拟的计算环境。

1. 第一类虚拟机管理程序

从技术上讲，第一类虚拟机管理程序就像一个操作系统，因为它是唯一——一个运行在最高特权的程序。它在裸机上运行并且具备多道程序功能。

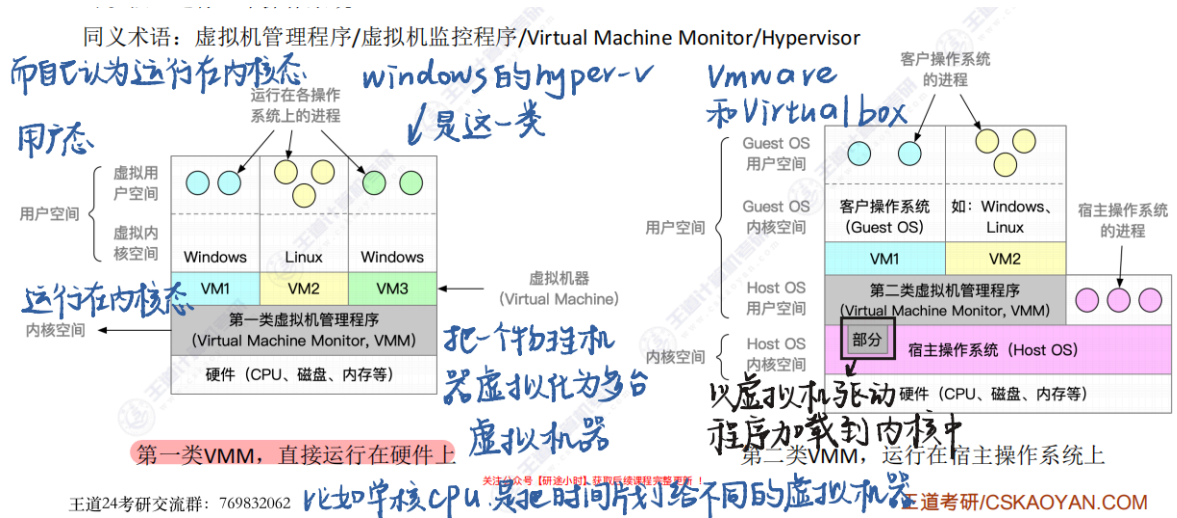
虚拟机管理程序向上层提供若干虚拟机，这些虚拟机是裸机硬件的精确复制品。因为每台虚拟机都与裸机相同，所以在不同的虚拟机上可以运行任何不同的操作系统。

2. 第二类虚拟机管理程序

它是一个依赖于 Windows、Linux 等操作系统分配和调度资源的程序，很像一个普通的进程。

第二类虚拟机管理程序仍然伪装成具有 CPU 和各种设备的完整计算机。

VMware Workstation 是首个x86 平台上的第二类虚拟机管理程序。

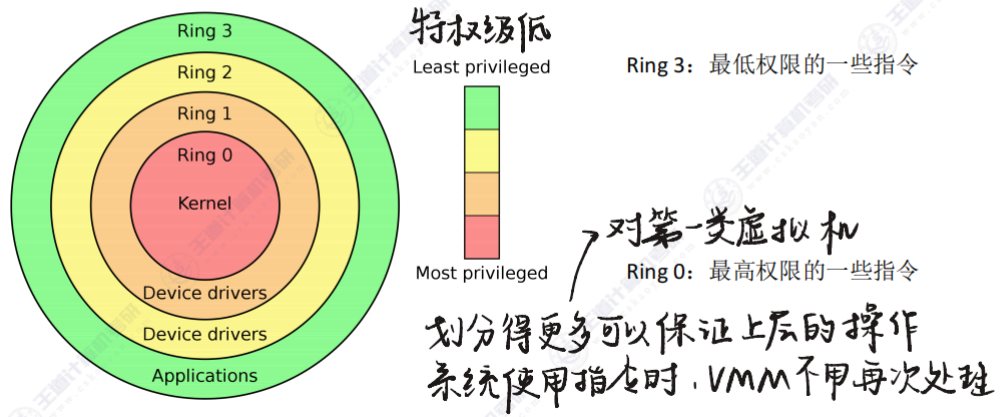


运行在两类虚拟机管理程序上的操作系统都称为客户操作系统。

对于第二类虚拟机管理程序，运行在底层硬件上的操作系统称为宿主操作系统。

两类虚拟机管理程序 (VMM) 的对比		Host OS 先给 VMM
	第一类VMM	第二类VMM
对物理资源的控制权	直接运行在硬件之上，能直接控制和分配物理资源	运行在Host OS之上，依赖于Host OS为其分配物理资源
资源分配方式	在安装Guest OS时，VMM要在原本的硬盘上自行分配存储空间，类似于“外核”的分配方式，分配未经抽象的物理硬件	GuestOS 拥有自己的虚拟磁盘，该盘实际上是Host OS 文件系统中的一个文件。GuestOS分配到的内存是虚拟内存
性能	性能更好	性能更差，需要HostOS作为“中介”
可支持的虚拟机数量	更多，不需要和 Host OS 竞争资源，相同的硬件资源可以支持更多的虚拟机	更少，Host OS 本身需要使用物理资源，Host OS 上运行的其他进程也需要物理资源
虚拟机的可迁移性	更差	更好，只需导出虚拟机镜像文件即可迁移到另一台 HostOS 上，商业化应用更广泛
运行模式	第一类VMM运行在最高特权级 (Ring 0)，可以执行最高特权的指令。	第二类VMM部分运行在用户态、部分运行在内核态。GuestOS 发出的系统调用会被 VMM 截获，并转化为 VMM 对 HostOS 的系统调用

支持虚拟化的CPU通常分更多指令等级



七、本章难点