第一章 计算机系统概述

2022年9月6日 星期二 16:45

操作系统的概念

- 操作系统负责管理协调硬件，软件等计算机资源的工作
- 操作系统为上层用户，应用程序提供简单易用的服务
- 操作系统是一种系统软件

操作系统的特征

- 并发和共享最基本的两个性质
- 并发和共享互为存在条件
- 没有并发和共享，就谈不上虚拟和异步

并发	• 并发：两个或多个事件在同一时间间隔内发送【同一时间间隔】 • 并行：系统具有同时进行运算或操作的特性【同一时刻】 ○ 可并行的有【处理机与设备】【处理机与通道】【设备与设备】 ○ 不可并行的有【进程与进程】 ○ 真正实现并行的是多核处理机
共享	• 共享是指系统中的资源可供内存中多个并发执行的程序共同使用 互斥共享方式：• A用完之后，才可以给B用；• 如对摄像头设备的共享使用 同时共享方式：• 一段时间内由多个进程同时访问；• 如对硬盘资源的共享使用
虚拟	• 虚拟是指把一个物理上的实体变为若干逻辑上的对应物 空分复用技术【虚拟的扩充空间】：• 如虚拟存储器 时分复用技术【虚拟的扩充时间】：• 如处理器的分时共享
异步	• 进程的执行不是一贯到底的，而是走走停停的，它以不可预知的速度向前推进

操作系统的功能和目标

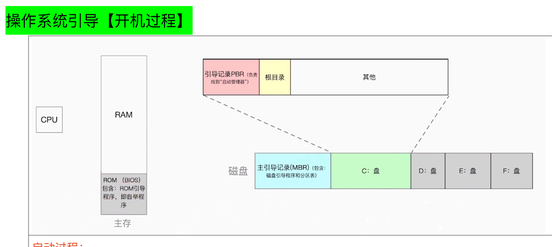
资源的管理者	• 处理器管理 ○ 处理机的分配和运行都以进程为基本单位，处理器管理=对进程的管理 ○ 主要功能：进程控制+进程同步+进程通信+死锁处理+处理机调度 • 存储器管理 ○ 为了给多道程序的运行提供良好的环境，方便用户使用及提高内存的利用率 ○ 主要功能：内存分配与回收+地址映射+内存保护+共享和内存扩充 • 文件管理 ○ 计算机的信息都是以文件的形式存在的 ○ 主要功能：文件存储空间的管理+目录管理+文件读写管理和保护 • 设备管理 ○ 主要是完成用户的I/O请求，方便用户使用各种设备，提高设备的利用率 ○ 主要功能：缓冲管理+设备分配+设备处理+虚拟设备
为用户和计算机硬件系统提供接口	• 给用户使用的 ○ GUI用户图像界面 ○ 命令接口 联机命令接口：【用户发送一个命令，系统就执行一次，主要特点是交互性，适用于分时或实时系统】 脱机命令接口：【用户一次性发送命令清单，系统按清单执行，中途不能干预，适用于批处理系统】【解决独占问题】 • 给软件/程序员使用的 ○ 程序接口【即系统调用】
对计算机资源的扩充	• 裸机：没有任何软件支持的计算机 • 扩充机器/虚拟机：覆盖了软件的机器

操作系统结构

- 微内核OS知识点补充
  - 优点：【内核足够小】【基于C/S模式】【应用机制与策略分离原理】【采用面向对象技术】
  - 缺点：【性能问题】【开销偏大】

	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	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	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	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	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）	1. 性能高，内核内部各种功能都可以直接相互调用	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，就可能导致整个系统崩溃
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	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	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	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	1. 降低了系统的一致性 2. 使系统变得更复杂

操作系统引导【开机过程】



启动过程：

1. CPU加电，CS：IP指向FFFF0H
2. 执行JMP指令跳转到BIOS
3. 登记BIOS中断例程入口地址
4. 硬件自检
5. 进行操作系统引导

引导过程

step1：CPU从一个特定主存地址开始，取指令，执行ROM中的引导程序【即前面的启动过程】

step2：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

step3：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

step4：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，完成开机的一系列操作

注意点：

- 引导程序有两种
  - 一种是位于ROM中的自举程序（BIOS的组成部分），用于启动具体的设备
  - 一种是位于装有操作系统硬盘的活动分区的引导扇区的引导程序（称为启动管理器），用于引导操作系统
- 操作系统被装入RAM中
- 自举程序BIOS装在ROM中
- 引导程序装在硬盘中

虚拟机

定义	• 使用虚拟化技术，将一台物理机器虚化为多台虚拟机器VM，每个虚拟机器都可用独立运行一个操作系统
分类	见下表

	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 文件系统中的一个大文件。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

操作系统发展历程

常考的三种操作系统对比

批操作系统	脱机使用计算机；作业是批处理的；系统内多道程序并发执行；交互能力差；
分时操作系统	多个用户同时使用计算机；人机交互强；具有每个用户独立使用计算机的独占性；系统响应及时
实时操作系统	能对控制对象做出及时反应；可靠性高；响应及时；但资源利用率低

其他操作系统对比

- Unix系统是多用户，多任务操作系统，属于分时操作系统

	定义	特点	优点	缺点
手工操作阶段				• 用户独占全机，资源利用率低 • CPU等待手工操作，CPU的利用不充分
单道批处理系统		• 自动性，顺序性，单道性		• 资源利用率低，吞吐量小
多道批处理系统	• 批处理：允许多个用户将若干个作业提交给计算机系统集中处理 • 多道性是为了提高系统利用率和吞吐量而提出的	• 多道，共享性，宏观上并行，微观上串行 • 引入多道后，系统就失去了封闭性和顺序性 • 制约性，程序执行因为共享资源和协同所有产生了竞争，相互制约 • 考虑到竞争的公平性，程序的执行是断续的	• 资源利用率高 • 系统吞吐量大 • CPU利用率高 • IO设备利用率高	• 用户响应时间长 • 不提供人机交互能力【批处理作业时用户无法干预】【主要缺点】
分时操作系统	• 允许多个用户以交互的方式使用计算机的操作系统 • 要求快速响应用户是导致分时系统出现的重要原因 • 响应时间 = 时间片*用户 • 进程调度通常采用抢占式的优先级高者优先	• 同时性 • 交互性 • 独立性 • 及时性	• 提供人机交互能力	• 不能再规定的时间内做出处理
实时操作系统	• 在该操作系统下，计算机系统能及时处理由过程控制反馈的数据，并及时做出响应	• 硬实时系统：必须在绝对严格的时间内完成处理。如导弹控制系统，自动驾驶系统 • 软实时系统：能偶尔接受违反时间规定，如12306火车订票系统	• 追求的目标：【安全可靠】【及时处理】【快速处理】	
分布式操作系统		• 分布性，并行性		
网络操作系统		• 网络中各种资源的关系及各台计算机之间的通信		

操作系统运行的环境

处理器运行的机制

程序运行的原理	○ 高级语言编写代码，然后转换为机器指令 ○ 程序运行的过程就是CPU执行指令的过程
两种程序	○ 内核程序 ○ 应用程序
两种指令	特权指令：指不允许用户直接使用的指令 → • 对I/O设备操作指令 • 存取特殊寄存器的指令 • 有关访问程序状态的指令 • 置中断指令 • 关中断指令 • 清内存指令 • 置时钟指令 非特权指令：允许用户直接使用的指令；不能直接访问系统中的软硬件资源；只限于访问用户的地址空间 → • 访管/trap指令
两种处理器状态	核心态【管态】：• 只能在核心态运行的指令和程序：• 时钟管理相关的指令【置时钟指令】• 中断机制相关的指令【时钟中断程序】• 原语相关的指令 • 系统控制的数据结构与处理【进程调度程序】【进程切换】【缺页处理程序】【系统调用命令】 用户态【目态】：• 在用户态运行的指令和程序/发生的事件：• 命令解释程序【属于命令接口，面向用户】• 访管/Trap指令，跳转指令，压栈指令 • 广义指令(系统调用)的调用 • 外部中断，缺页
如何变态	内核态--- >用户态：发送在中断返回用户程序是，需要一条修改PSW(程序状态字)的特权命令 用户态---->内核态：发送在中断时，通过硬件完成
为什么要区别核心态和用户态?	为了保护系统程序

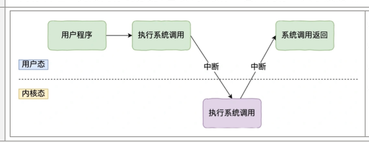
内核的功能

- 时钟管理：实现计时功能
- 中断处理：负责实现中断机制
- 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，创建和删除等功能
- 进程管理：完成进程的创建，撤销，阻塞及唤醒等功能
- 进程通信：完成进程之间的信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大小及地址等功能
- 原语
  - 是一种特殊的程序
  - 处于操作系统最底层，是最接近硬件的部分
  - 该程序运行具有原子性（运行只能一气呵成，不可中断）
  - 运行时间较短，调用频繁

外中断和内中断（异常和中断控制【结合OS学习】）

	异常（内中断）	中断（外中断）
基本概念	• 由CPU内部产生的意外事件 • 是CPU执行一条命令时，由CPU在其内部检测到的、与正在执行指令相关的同步事件 • 故障和自陷为异常 • 终止异常和外中断属于硬件中断	• 由来自CPU外部的设备发出的中断请求（常用于输入输出） • 典型的由外部设备触发的、与当前正在执行的指令无关的异步事件 • 外部I/O设备通过特定的中断请求信号线向CPU提出中断请求 • CPU每执行完一条指令就检查中断请求信号线，若检测到中断请求，则进入中断响应期 • 外部中断都是在一条指令执行完成后（中断周期）才被检测并处理的
分类	故障：• 在引起故障的指令启动之后、执行结束前被检测到的异常事件 自陷：• 也称陷阱或陷入，是预先安排的一种"异常事件"，就像预先设置好的"陷阱"一样 终止：• 若在执行指令的过程中发生了使计算机无法继续执行的硬件故障，那么程序将无法继续执行，只能终止	可屏蔽中断：• 通过可屏蔽中断请求线INTR向CPU发出的中断请求 • CPU可以通过在中断控制器中设置相应的屏蔽字来屏蔽或不屏蔽它，被屏蔽的中断信号将不被送到CPU 不可屏蔽中断：• 通过不可屏蔽中断请求线NMI向CPU发出的中断请求 • 通常是非常紧急的硬件故障，如电源掉电等。
举例	故障：• 指令译码时，出现"非法操作码" • 取数据时，发生"缺段"或"缺页" • 除数为零 • 地址越界 自陷：• x86机器中，用于程序调试"断点设置"和单步跟踪功能 • 系统调用指令 • 条件自陷指令 终止：• 控制器出错 • 存储器校验错 • 调出中断服务程序来重启系统	• Cache缺失 • I/O中断：键盘输入，打印机缺纸 • 时钟中断 • I/O中断请求
执行时间	• 每个指令周期末尾，CPU都会检查是否有外中断信号需要处理	• CPU在执行指令时会检查是否有异常发生
不同点	• 缺页"或"溢出"等异常事件是由特定指令在执行过程中产生的 • 异常的检测由CPU自身完成，不必通过外部的某个信号通知CPU	• 中断不与任何指令相关联，也不阻止任何指令的完成

系统调用【又叫做广义指令】

定义	○ 操作系统对应应用程序/程序员提供的接口 ○ 系统调用需要触发Trap【也叫陷入/访管指令】 ○ os通过提供系统调用避免用户程序直接访问外设【应用程序通过系统调用使用OS的设备管理服务】
目的	请求系统服务
与库函数的区别	库函数：• 是语言或应用程序的一部分，可以运行在用户空间中 • 许多库函数都会使用系统调用来实现功能 • 有的库函数没有使用系统调用 系统调用：• 是操作系统的一部分，是内核为用户提供的程序接口，运行在内核空间 • 未使用系统调用的库函数，执行效率通常比系统调用的高【因为系统调用要完成上下文的切换和状态的转换】
按功能分类	○ 设备管理：完成设备的请求或释放+设备启动 ○ 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 ○ 进程通信：完成进程之间的信息传递或信号传递 ○ 内存管理：完成内存的分配+回收+获取作业占用内存区大小及始址
系统调用的过程	 step1：传参 step2：陷入指令/Trap/访管【执行系统调用】 step3：由操作系统内核程序处理系统调用请求 step4：返回应用程序