操作系统
Operating System

北京理工大学计算机学院
马　锐
Email: mary@bit.edu.cn

版权声明

第1章　操作系统概论

1.1 操作系统的定义

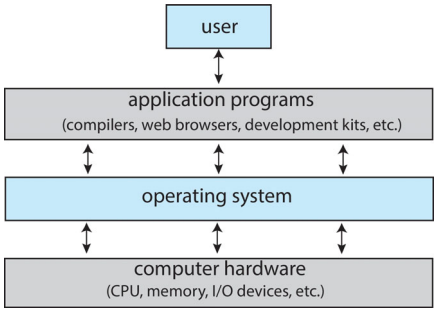
1.1.1 计算机系统的组成

1.1.2 操作系统的作用

1.1.3 操作系统的定义

4

1.1.1 计算机系统的组成



Abstract view of the components of a computer system[1].

5

1.1.2 操作系统的作用

- 用户角度
  - 作为用户与计算机硬件系统之间的接口
  - 设计目标
    - 有效性、方便性
- 计算机系统设计者角度
  - 作为计算机系统资源的管理者
  - 设计目标
    - 提高资源利用率

6

1.1.3 操作系统的定义(1)

- 操作系统是计算机系统中的一个系统软件，是一些程序模块的集合
  - 它们能以尽量有效、合理的方式组织和管理计算机的软、硬件资源，合理的组织计算机的工作流程，控制程序的执行并向用户提供各种服务功能，使得用户能够灵活、方便、有效的使用计算机，使整个计算机系统能高效地运行。是计算机与用户之间的接口

7

1.1.3 操作系统的定义(2)

- "The one program running at all times on the computer" is the kernel, part of the operating system
- Everything else is either
  - a system program (ships with the operating system, but not part of the kernel) , or
  - an application program, all programs not associated with the operating system
- Today's OSes for general purpose and mobile computing also include middleware – a set of software frameworks that provide addition services to application developers such as databases, multimedia, graphics

8

1.2 操作系统的形成与发展

1.2.1 无操作系统的计算机系统
1.2.2 单道批处理操作系统
1.2.3 多道批处理操作系统
1.2.4 分时与多任务操作系统
1.2.5 实时操作系统
1.2.6 嵌入式操作系统
1.2.7 智能移动终端操作系统
1.2.8 分布式系统
1.2.9 国产操作系统
1.2.10 操作系统的发展趋势

9

1.2 操作系统的形成与发展

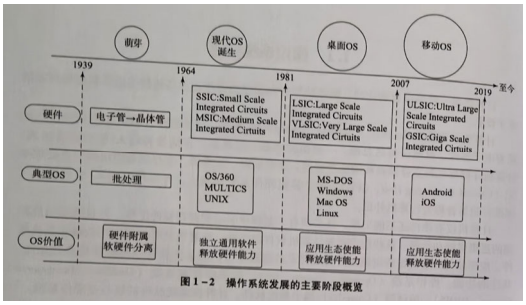


图1-2 操作系统发展的主要阶段概览

10

1.2.1 无OS的计算机系统

- 程序员直接使用计算机硬件系统，效率低下
- 单用户独占全机
- CPU等待人工操作
  - 人工负责输入输出
  - 人工负责计算机的调度
  - 人工负责编排作业的运行顺序

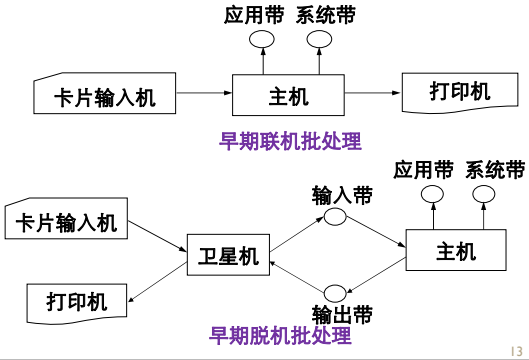
11

1.2.2 单道批处理操作系统(1)

- 以常驻内存的监控程序代替人工调度和作业编排，减少人工干预和等待时间，加快作业运行速度
- 硬件结构
  - 早期联机批处理
    - 作业的输入、计算和输出都在CPU控制下进行，CPU利用率低
  - 早期脱机批处理
    - 小型卫星机控制外部设备的输入和输出

12

1.2.2 单道批处理操作系统(2)



13

1.2.2 单道批处理操作系统(3)

- 优点
  - 系统自动化程度高、吞吐量大，资源利用率高
  - 减少了CPU的空闲时间
  - 提高了I/O速度
- 缺点
  - CPU与外设串行
  - 作业周转时间长，用户无法实现对作业的控制

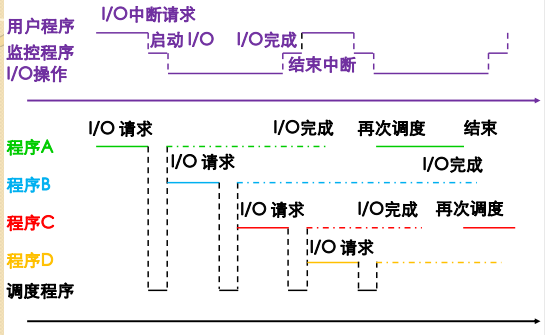
14

1.2.3 多道批处理操作系统(1)

- 多道程序设计
  - 引入原因：CPU与外设并行执行，提高CPU利用率
  - 在内存中同时存放若干道程序，使之在系统中同时处于交叉运行状态
  - 特点
    - 内存多道
    - 宏观上并行（不同的作业分别在CPU和外设上执行）
    - 微观上串行（在CPU上交叉运行）

15

1.2.3 多道批处理操作系统(2)



单道程序与多道程序示意图

16

1.2.3 多道批处理操作系统(3)

- 目的
  - 提高CPU的利用率
  - 充分发挥系统设备的并行性
    - 程序之间
    - CPU与设备之间
    - 设备与设备之间

➢ 单道程序设计示例：主存中有一道程序，读写一个记录各需0.0015s，处理一个记录（执行100条指令）需0.0001s，则CPU利用率为

0.0001/(0.0015+0.0001+0.0015) ≈3.2%

17

1.2.3 多道批处理操作系统(4)

➢ 多道程序设计示例：若一个计算机系统有256K内存(不包含OS)，1个磁盘、1个终端和1台打印机。内存中装有的3个作业对资源的使用情况如下：

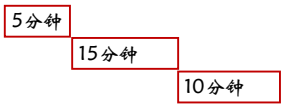
作业编号	JOB1	JOB2	JOB3
类型	计算型	I/O型	I/O型
占用内存	50K	100K	80K
使用磁盘	NO	NO	YES
使用终端	NO	YES	NO
使用打印机	NO	NO	YES
运行时间	5分钟	15分钟	10分钟

18

1.2.3 多道批处理操作系统(5)

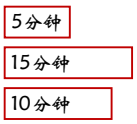
单道批处理

- ✓ 作业1运行5分钟
- ✓ 作业2等待5分钟运行15分钟
- ✓ 作业3等待20分钟运行10分钟



多道批处理

- ✓ 三个作业同时装入主存，由于几乎不同时使用同类资源，在15分钟内将全部完成



19

1.2.3 多道批处理操作系统(6)

单道/多道程序设计方式下的资源利用率

	单　道	多道（三道）
CPU利用率	17%=5/30	33%=5/15
存储器利用率	30%=(50/256+100/256+80/256)/3	90%=230/256
磁盘利用率	33%=10/30	67%=10/15
打印机利用率	33%=10/30	67%=10/15
全部完成时间	30min.	15min.
吞吐量	6jobs/hour=3/0.5	12jobs/hour=3/0.25
平均周转时间	18min.=(5+20+30)/3	10min.=(5+15+10)/3

20

1.2.3 多道批处理操作系统(7)

- ➢ 衡量批处理系统的性能指标
  - 资源利用率：给定时间内系统中某一资源(存储器、CPU、外设等)实际使用时间所占比率。
  - 吞吐量：单位时间内系统所处理的信息量，通常以每小时或每天所处理的作业个数进行度量。
  - 周转时间：从作业进入系统到作业退出系统(即完成)所用的时间。
  - 平均周转时间：系统运行的多个作业的周转时间的平均值。

21

1.2.3 多道批处理操作系统(8)

- 多道批处理系统的特点
  - 有效地提高了系统资源的利用效率
  - 提高了系统的吞吐量
  - 用户与作业之间无法交互
  - 作业平均周转时间较长
  - 适合处理计算量大的成熟的作业
- 典型系统
  - IBM OS/360：首次将操作系统与计算机相分离，从专用走向通用的操作系统

22

1.2.4 分时与多任务操作系统(1)

- 批处理系统的缺点
  - 用户不能直接控制作业运行
  - 作业周转时间太长
- 产生分时系统的原因
  - 用户需求（交互+响应）
- 分时概念
  - 多个用户分时使用CPU时间，即将CPU的单位时间划分成若干时间段（每个时间段称为一个时间片），各用户按时间片轮流占用CPU

23

1.2.4 分时与多任务操作系统(2)

- 分时系统的特点
  - 同时性
  - 独立性
  - 交互性
  - 及时性
    - 响应时间为2-3s
    - 响应时间是衡量分时系统的主要指标
    - 影响响应时间的因素
      - 用户数目
      - 时间片

24

1.2.4 分时与多任务操作系统(3)

批处理系统	分时系统
➢处理对象为作业	➢处理对象为作业
➢目标是提高系统资源利用率	➢目标是对用户请求快速响应
➢适用于比较成熟的大型作业	➢适用于短小作业
➢可在后台执行，不需要用户频繁干预	➢终端键入命令

25

1.2.4 分时与多任务操作系统(4)

- 典型系统
  - 第1个分时系统CTSS(Compatible Time-Sharing System)
  - 典型的具有重要意义的分时系统MULTICS(Multiplexed Information and Computing System)
  - UNIX / LINUX

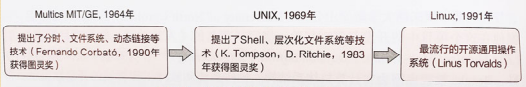


图 1-2 分时与多任务操作系统的演进

1.2.4 分时与多任务操作系统(5)

- DOS/Windows
- macOS

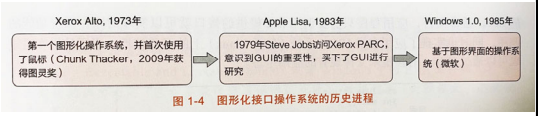


图 1-4 图形化接口操作系统的历史进程

27

1.2.5 实时系统(1)

- 实时概念
  - 系统对外部特定输入信号的响应时间足以控制发出实时信号的设备
- 实时系统分类
  - 应用需求
    - 实时控制系统/实时信息处理系统
  - 实时任务的截止时间
    - 硬实时/软实时

28

1.2.5 实时系统(2)

- 实时系统的特点
  - 实时性
  - 可靠性
  - 可确定性
- 实时系统适合于一些不强调资源利用效率的专用系统
- 实时系统以数据或信息作为处理对象

29

1.2.6 嵌入式操作系统

- 嵌入式操作系统
  - 以应用为中心、以计算机技术为基础、软硬件可裁剪的专用计算机系统
  - 基本都是实时操作系统，只有满足实际需要的有限功能，如任务调度、同步与通信、主存管理、时钟管理等
  - 特点
    - 软件要求固化存储
    - 高质量、高可靠性、高实时性
  - 典型系统
    - Linux，FreeRTOS

30

1.2.7 智能移动终端操作系统

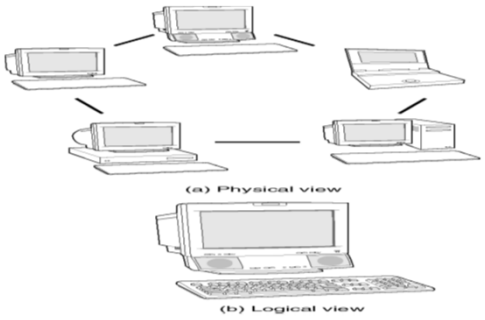
- 智能移动终端操作系统
  - 在受限的计算能力上提供丰富的用户体验
  - 在受限的供电能力上改善系统的能源效率
  - 在存储大量用户敏感信息场景下保证用户信息的隐私和安全
  - 典型系统
    - Android
    - iOS
    - Harmony

31

1.2.8 分布式系统(1)

- Definition of a Distributed System
  - A distributed system is a collection of independent computers appears to its users as a single coherent system.
    - Component: autonomous
    - User: a single system
  - A distributed system is the one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.

1.2.8 分布式系统(2)



1.2.9 国产操作系统(1)

- 国产操作系统的发展——启蒙阶段
  - 国产操作系统主要以概念学习与初步尝试为主，停留在实验室阶段
  - 1983年8月底，电子工业部6所，中国第一款自主研发的操作系统CCDOS。CCDOS的基础是微软的DOS，在中国PC发展史上有程碑式的意义
  - “八五”计划提出打造我国自主版权操作系统，制定了以UNIX为技术路线的计划
  - 在1989年，中软开发了中国第一款通用操作系统COSIX1.0，之后在1994-1995年推出COSIX V2.0操作系统

34

1.2.9 国产操作系统(2)

- 国产操作系统的发展——发展阶段
  - 国产操作系统渐渐步入实用化阶段
  - 1999年4月8日，中国第一款基于Linux/Fedora的国产操作系统Xteam Linux 1.0 发布，开启了新阶段的操作系统国产化之路。蓝点、红旗、中软三足鼎立，分别推出Linux产品
    - 1999年7月，蓝点（BluePoint） Linux中文版（预览版）横空出世
    - 1999年8月，以中科院院士倪光南、中科院软件研究所副所长孙玉芳为首的一批科学家，推出国产操作系统-红旗Linux
    - 1999年9月，中国软件总公司第一个中文Linux版本发布
  - 1999年也称为基于Linux内核的国产操作系统元年

35

1.2.9 国产操作系统(3)

- 国产操作系统的发展——壮大阶段
  - 更多成熟的国产操作系统诞生
  - 2001年，国家863计划重大攻关科研项目支持的银河麒麟（Kylin）操作系统诞生，但体验感较差
  - 2006年12月4日，中国科学技术部宣布，研制成功当时中国通过认证的安全等级最高、拥有自主知识产权的操作系统——“银河麒麟”服务器操作系统，成为国产操作系统发展中的重要里程碑
  - 2010年12月，民用“中标 Linux”与“银河麒麟”正式在上海合并为中标麒麟 Neokylin。中标麒麟系统在我国国防、航天、电力、能源、政务等众多重要行业得到广泛应用，并多年成为我国Linux市场占有率第一的操作系统
  - 2010年，华为在公司内部发布服务器操作系统EulerOS，开始在公司内部的云产品及ICT产品规模化使用

36

1.2.9 国产操作系统(4)

- 国产操作系统的发展——攻坚阶段
  - 国产操作系统经历多代技术迭代，软件生态持续壮大
  - Deepin深度操作系统：累计下载超过8000万次，成为中国民用市场最为成功的本土化桌面操作系统之一
  - UOS是由统信软件开发的一款基于Linux内核的操作系统，可以说其脱胎于深度操作系统，主要分为统一桌面操作系统和统一服务器操作系统
  - OpenEuler是面向数字基础设施的操作系统，是华为公司基于CentOS制作的Linux发行版，支持服务器、云计算、边缘计算、嵌入式等应用场景，支持多样性计算，致力于提供安全、稳定、易用的操作系统
  - OpenHarmony

37

1.2.9 国产操作系统(5)

- 华为鸿蒙系操作系统HarmonyOS
  - 华为自主设计研发的国产操作系统
  - 是一款全新的面向未来、面向全场景的分布式操作系统
  - 创造一个超级虚拟终端互联的世界，将人、设备、场景有机地联系在一起，将消费者在全场景生活中接触的多种智能终端实现极速发现、极速连接、硬件互助、资源共享，用合适的设备提供场景体验

38

1.2.9 国产操作系统(6)

- HarmonyOS发展历程
  - 2019年8月9日，华为开发者大会上，HarmonyOS 1.0正式面世，是一款全新的面向全场景的分布式OS（宣布开源）
  - 2020年9月10日，华为正式发布HarmonyOS 2.0商用版本，成功在智能手机、平板电脑和智能穿戴设备等多个设备上首次应用，实现了“1+8+N”全场景战略
  - 2021年9月23日晚间，华为官宣鸿蒙系统升级用户已经突破1.2亿，平均每天超100万用户升级鸿蒙，已经成为迄今全球用户增长速度最快的移动操作系统

39

1.2.9 国产操作系统(7)

- HarmonyOS发展历程
  - 2022年7月27日，华为正式发布鸿蒙3（HarmonyOS 3）操作系统
  - 2023年8月4日，华为终端BG CEO余承东在华为终端开发者大会上正式宣布，华为鸿蒙4（HarmonyOS 4）操作系统正式发布
  - 2023年8月4日，华为公司自研操作系统HarmonyOS NEXT，是鸿蒙抛弃Linux内核及安卓开放源码项目等代码后的收个大版本，不再兼容安卓应用

40

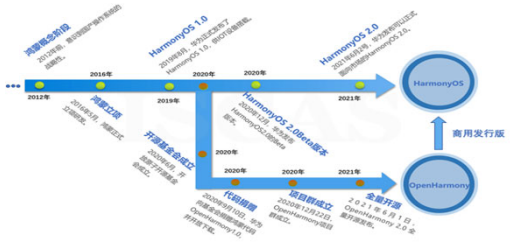
1.2.9 国产操作系统(8)

- HarmonyOS与OpenHarmony
  - 开放原子开源基金会(Openatom Foundation)
    - 中国第一家开源基金会，旨在推动全球范围科技与产业的发展
    - 该基金会的成立标志着中国在开源领域取得了重要进展，并得到了多家龙头科技企业的联合发起支持
    - 开源鸿蒙项目是开放原子开源基金会正在孵化的项目之一。在这个项目中，华为将HarmonyOS的基础能力捐赠给了开源基金会，形成了OpenHarmony的初始版本。而HarmonyOS则是基于OpenHarmony的首个商业发行版本

41

1.2.9 国产操作系统(9)

- HarmonyOS与OpenHarmony



42

1.2.9 国产操作系统(10)

国家首个开放原子开源基金会

从2020 年开始，华为持续将自研终端操作系统的基础能力贡献给开放原子开源基金会。
由基金会整合其他参与者的贡献，形成OpenHarmony开源项目。

OpenHarmony开源项目历程

开放原子开源基金会官网：https://www.openatom.org
代码仓库：https://openharmony.gitee.com

2020年9月10日
- 开放原子开源基金会正式上线 OpenHarmony 1.0版本；
- 支持128K到128M的智能终端

2020年12月
- 七家单位成立 OpenHarmony 项目群工作委员会（博泰、华为、京东、润和、亿咖通、中科院软件所、中软国际）；
- 华为是开源项目贡献者之一，也是开源版本的使用者之一

2021年6月1日
- 上线 OpenHarmony 2.0 Canary 版本；
- 支持内存128M 以上的智能终端

2021年9月30日
- 上线 OpenHarmony 3.0 LTS 版本；
- 适配多款开发板，发布3000 JS APIs，可支撑10+类IoT 产品商用

……

- 开放原子开源基金会，是国内首个/当前唯一的开源基金会，由工信部指导成立。
- 在海外，2021年Eclipse 基金会宣布启动 Oniro 项目和工作组，该项目和工作组立志成为已建立的物联网和边缘操作系统的透明、供应商中立和独立的替代方案。Oniro作为首个OpenHarmony的全球版本，将可用于各种平台，包括移动端、物联网、工业设备及等其他相关设备。

1.2.9 国产操作系统(11)

OpenHarmony1.0.1 LTS　OpenHarmony2.0 LTS　OpenHarmony3.0/3.2 LTS　OpenHarmony4.0 LTS 2023年11月　OpenHarmony X.0 LTS

OpenHarmonyLTS版本的发展历程

1.2.9 国产操作系统(12)

- OpenHarmony架构

1.2.9 国产操作系统(13)

- OpenHarmony架构
  - 内核层
    - 多内核设计
    - 内核子系统+驱动子系统
  - 系统服务层
    - HarmonyOS的核心能力集合
    - 为应用系统开发提供了服务调用功能
  - 框架层
    - 为HarmonyOS的应用开发提供了不同语言程序调用的接口

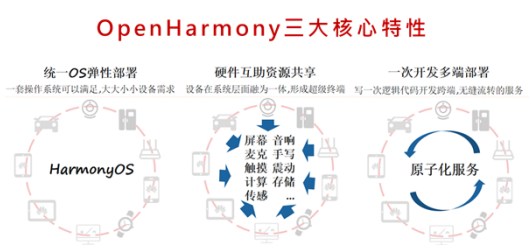
46

1.2.9 国产操作系统(14)

- OpenHarmony特点
  - 分布式架构
    - 多设备之间的连接和协同工作
  - 统一开发平台
    - 提供统一开发平台与工具，简化应用开发的流程
  - 高效性能
    - 支持异构多核技术，充分利用多核处理器的性能优势
  - 面向全场景
    - 适用多种终端设备，提供流畅的全场景体验
  - 保护安全隐私
    - 采用安全内核、安全通信、安全存储，保护安全隐私

47

1.2.9 国产操作系统(15)



48

1.2.9 国产操作系统(16)

- 硬件互助、资源共享
  - 分布式软总线
    - 解决1+8+N设备间互联问题
  - 分布式设备虚拟化
    - 多种设备共同形成一 个超级虚拟终端，针对不同类型任务匹配恰当硬件，发挥不同设备资源优势
  - 分布式数据管理
    - 用户数据不再与单 一物理设备绑定
    - 业务逻辑与数据存 储分离
    - 应用跨设备运行时数据无缝衔接
  - 分布式任务调度
    - 统一的分布式服务管理（发现、同步、注册、 调用）机制
    - 支持跨设备的应用进行 远程启动、远程调用、 远程连接以及迁移等操作

1.2.9 国产操作系统(17)

- 一次开发，多端部署
  - 提供了用户程序框架、Ability框架以及UI框架，在不同设备间迁移，并完成自适应布局
  - 实现同一应用一次开发，多端部署
  - 开发者只需关注业务逻辑，提升代码复用率
- 统一OS，弹性部署
  - 通过组件化和小型化等设计方法，支持多种终端设备按需弹性部署，能够适配不同类别的硬件资源和功能需求。
  - 支撑通过编译链关系去自动生成组件化的依赖关系，形成组件树依赖图，支撑产品系统的便捷开发，降低硬件设备的开发门槛

1.2.9 国产操作系统(18)

- 分布式安全
  - 确保正确的人、用正确的设备、正确使用数据
  - 当用户进行解锁、付款、登陆等行为时系统会主动拉出认证请求，并通过分布式技术可信互联能力，协同身份认证确保正确的人
  - 能够把手机的内核级安全能力扩展到其他终端，进而提升全场景设备的安全性，通过设备能力互助，共同抵御攻击，保障智能家居网络安全
  - 通过定义数据和设备的安全级别，对数据和设备都进行分类分级保护，确保数据流通安全可信

1.2.10 操作系统的发展趋势

- 操作系统与网络高度融合
  - 端、边、云的OS架构和关键技术将越来越趋同
- 一种设备类型一个OS的情况将难以为继
  - 统一OS将装载在各种形态的硬件设备商，OS不但与硬件充分解耦，与生态和应用充分解耦，其自身也将充分解耦
- OS可以调度所有连接在一起的硬件资源，将彻底推动OS计算调度机制、存储层次的改变
- OS将成为物理世界与数字世界融合的底座

52

1.3 操作系统的基本特性

- 并发性
  - 并发性：两个或多个事件在同一时间间隔内发生，它们都已经被启动执行，而且都还没有完成执行
  - 并行性：两个或多个事件在同一时刻发生
- 共享性
  - 互斥共享
- 虚拟性
- 异步性

53

1.4 操作系统的主要功能

- 处理机管理
  - 进程控制/进程同步/进程通信/进程调度
- 存储器管理
  - 内存分配/内存保护/地址映射/内存扩充
- 文件管理
  - 文件存储空间的管理/目录管理/文件的读写管理和保护
- 设备管理
  - 缓冲管理/设备分配

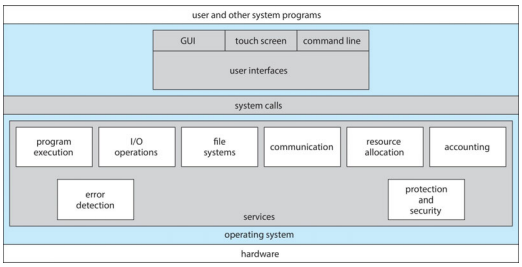
54

1.5 操作系统提供的服务(1)

- 用户接口
  - 命令级接口：Graphical User Interface/Touch Screen Interface/Command-Line Interface
  - 程序级接口（也称系统调用）
- 执行程序
- I/O操作
- 文件系统操作
- 通信服务：共享内存/消息传递
- 错误检测和处理
- 资源分配
- 登录和记账
- 保护和安全

55

1.5 操作系统提供的服务(2)



操作系统提供的服务

56

1.6 操作系统的用户接口(1)

- 操作接口
  - 命令行（命令解释程序）
    - OS内核的一部分(DOS)
    - 特殊程序，任务开始或用户登录时，该程序运行(UNIX的Shell)
    - 作用：执行命令
      - 命令解释程序执行(DOS)
      - 系统程序实现(UNIX的Shell)
  - Graphical User Interface
    - 具有窗口界面的解释程序(Windows的Explorer.exe)
  - Touch Screen Interface
    - Gesture, Voice Command

57

1.6 操作系统的用户接口(2)

- 编程接口
  - 系统调用
    - 请求操作系统服务或资源
  - 常以API形式出现
    - 适用于Windows系统的Win32 API
    - 适用于POSIX系统的POSIX API
      - POSIX: Portable Operating System Interface of UNIX
    - 设计运行于Java虚拟机的Java API
  - 应用领域接口
    - Android/iOS应用接口、汽车领域AUTOSAR

58

1.6 操作系统的用户接口(3)

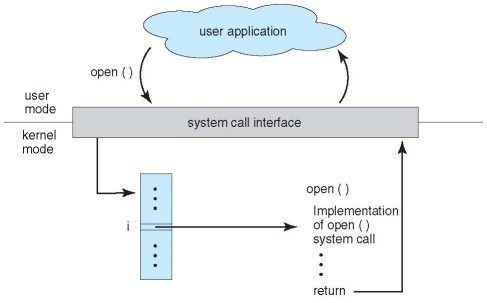
EXAMPLES OF WINDOWS AND UNIX SYSTEM CALLS

The following illustrates various equivalent system calls for Windows and UNIX operating systems.

	Windows	Unix
Process control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File management	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device management	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communications	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shm_open() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

59

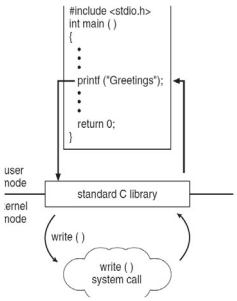
1.6 操作系统的用户接口(4)



系统调用与操作系统的关系

60

1.6 操作系统的用户接口(5)



标准C程序库示例

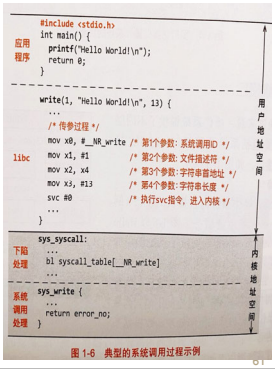


图 1-6 典型的系统调用过程示例

1.7 操作系统的运行方式(1)

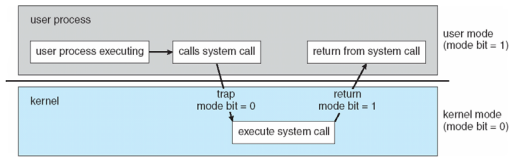
- 双重模式操作
  - 内核模式 — OS内核程序
    - 允许执行全部指令
    - 允许访问所有的寄存器和缓冲区
  - 用户模式 — 用户程序 系统外层应用程序
    - 只能执行非特权指令
    - 只能访问指定的寄存器和存储区

1.7 操作系统的运行方式(2)

- 特权指令
  - 能引起损害的机器指令
  - E.g. I/O控制，定时器管理，中断管理
    - 有关对I/O设备使用的指令：启动I/O设备指令、测试I/O设备工作状态和控制I/O设备动作的指令等
    - 有关访问程序状态的指令：操作程序状态字（PSW）的指令等
    - 存取特殊寄存器指令：存取中断寄存器、时钟寄存器等指令

1.7 操作系统的运行方式(3)

- 计算机硬件提供模式位



用户模式到内核模式的转换

64

1.7 操作系统的运行方式(4)

- 两种模式的转换
  - 用户模式--->内核模式
    - 中断、异常、系统调用
  - 内核模式--->用户模式
    - 完成中断或异常的处理
    - 新进程或线程启动
    - 进程调度选择执行新的进程
    - 操作系统向进程发送信号

65

1.8 操作系统的结构设计

1.8.1 操作系统的机制与策略
1.8.2 操作系统复杂度管理方法
1.8.3 操作系统内核架构
1.8.4 操作系统框架

66

1.8.1 操作系统的机制与策略

- 目的：降低操作系统设计的复杂性
- 设计原则：策略与机制分离
  - 策略(Policy)：做什么
  - 机制(Mechanism)：怎么做
  - 操作系统可仅通过调整策略来适应不同应用的需求
- 示例：调度
  - 策略：调度算法，FCFS，RR
  - 机制：调度队列的设计、调度实体（线程）的表示

67

1.8.2 操作系统复杂度管理方法

- 模块化(Modularity)
  - 分而治之
- 抽象(Abstract)
  - 接口与内部实现分离
- 分层(Layering)
  - 按层次划分，减少模块之间的交互
- 层级(Hierarchy)
  - 将功能相似的子系统组成一个具有清晰接口的自包含子系统，子系统递归形成大系统

68

1.8.3 操作系统内核架构(1)

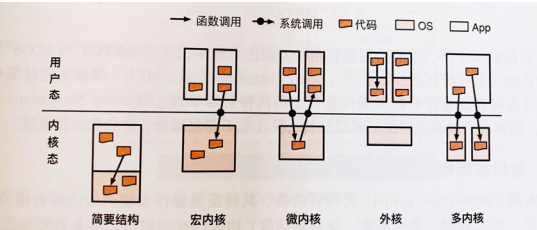


图 3-2 操作系统内核架构的频谱：简要结构（如 DOS）、宏内核（如 UNIX/Linux）、微内核、外核与多内核等

69

1.8.3 操作系统内核架构(2)

- 1.8.3.1 简要结构
- 1.8.3.2 宏内核架构(Monolithic Kernel)
- 1.8.3.3 微内核架构(Microkernel)
- 1.8.3.4 外核架构(Exokernel)
- 1.8.3.5 多内核架构(Multikernel)
- 1.8.3.6 混合内核架构(Hybrid Kernel)

70

1.8.3.1 简要结构(1)

- 功能简单的操作系统，将应用程序和操作系统放置在同一个地址空间中，无须底层硬件提供复杂的内存管理、特权级隔离等功能
- 模块化与层次结构
  - 将操作系统按其功能划分为若干个具有一定独立性和大小的模块和子模块
  - 各模块间通过接口实现交互
    - 模块间调用关系无序
    - 模块间耦合紧密

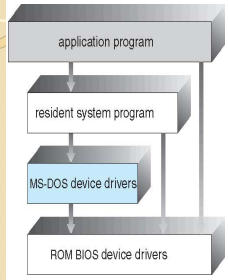
71

1.8.1 简要结构(2)

- 优点
  - 提高了OS设计的正确性、可理解性和可维护性
  - 增强了OS的适应性
  - 加速了OS的开发过程
- 缺点
  - 系统结构不清晰
  - 系统可靠性降低
  - 未能区分共享资源和独占资源

72

1.8.1 简要结构(3)



MS-DOS结构

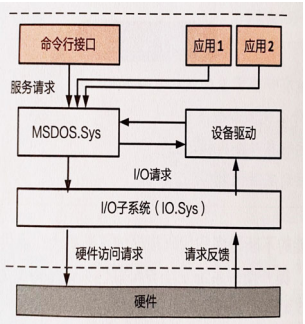


图 3-3 MS-DOS 的系统结构

73

1.8.3.2 宏内核架构(1)

- 操作系统内核的所有模块均运行在内核态（单内核），具备直接操作硬件的能力
- UNIX/LINUX，FreeBSD等

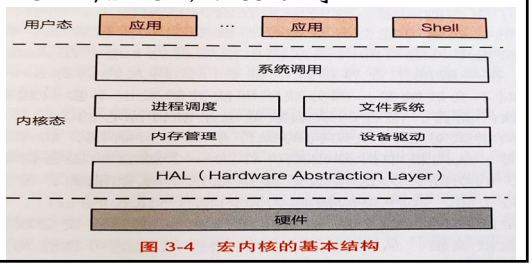
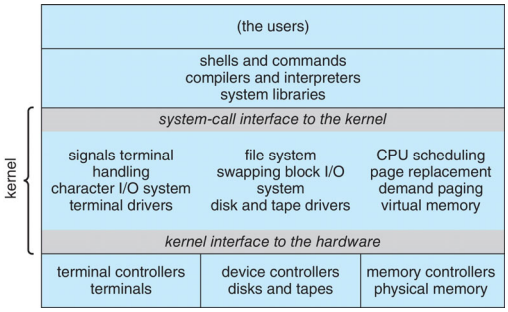


图 3-4 宏内核的基本结构

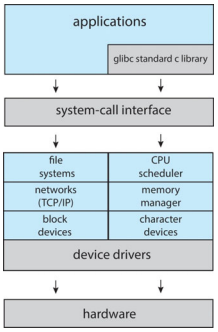
1.8.3.2 宏内核架构(2)



Traditional UNIX system structure.

75

1.8.3.2 宏内核架构(3)



Linux system structure.

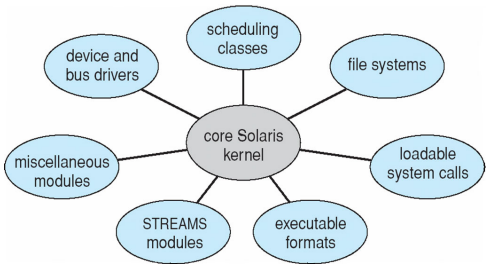
76

1.8.3.2 宏内核架构(4)

- 模块化
  - 内核有一组核心部件，以及在启动或运行时对附加服务的动态链接
  - 内核提供核心服务，并能动态实现特定的功能
  - 可加载内核模块LKM(Loaded Kernel Module)机制
  - 现代操作系统广为采用
    - UNIX/Linux/Windows等

77

1.8.3.2 宏内核架构(5)



Solaris结构

78

1.8.3.2 宏内核架构(6)

- 抽象
  - UNIX：一切皆是文件(Everything is a file.)，将数据、设备、内核对象等均抽象为文件，并为上层提供统一接口。
- 分层
  - 宏内核架构的操作系统一开始就采用了分层结构
  - 引入原因
    - 模块间有序调用
  - 设计原则
    - 将功能模块排列成若干层

79

1.8.3.2 宏内核架构(7)

- 各层之间的模块只能单向调用，每一层都仅使用其底层（或内层）模块所提供的功能和服务
- 每一层的同层模块之间不存在相互调用关系
- 最底层（第0层）是计算机硬件；最高层（第N层）是用户接口
- 优点
  - 模块化
  - 易于实现
  - 增加了系统的可靠性

80

1.8.3.2 宏内核架构(8)

- 示例：1972年图灵奖获得者Edsger Dijkstra提出“THE”操作系统

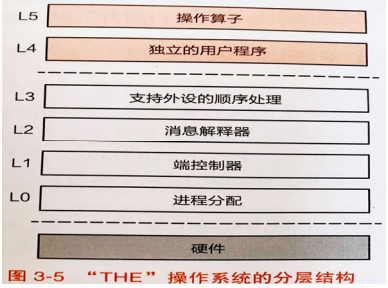


图 3-5 “THE”操作系统的分层结构

81

1.8.3.2 宏内核架构(9)

- 示例：Linux的文件系统

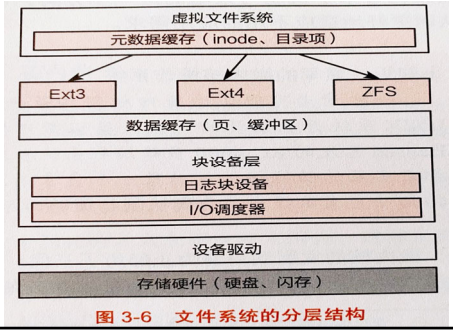
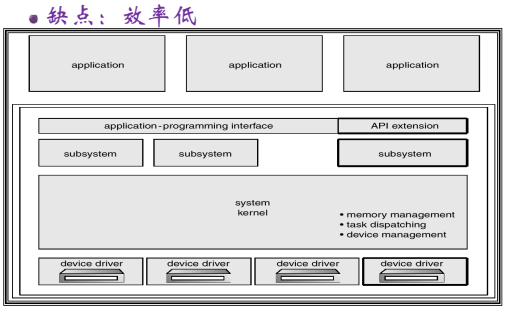


图 3-6 文件系统的分层结构

82

1.8.3.2 宏内核架构(10)

- 困难：难以确切地定义每一层
- 缺点：效率低

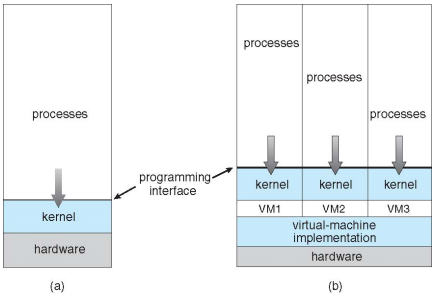


OS/2分层结构

83

1.8.3.2 宏内核架构(11)

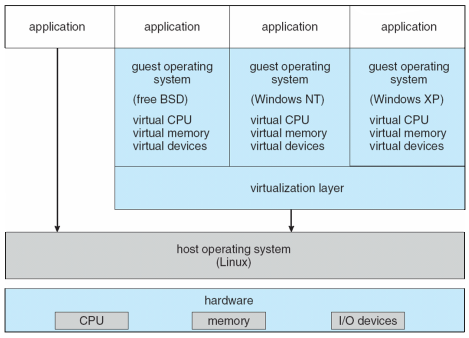
➢ 虚拟机



非虚拟机与虚拟机系统模型

84

1.8.3.2 宏内核架构(12)



VMware结构

85

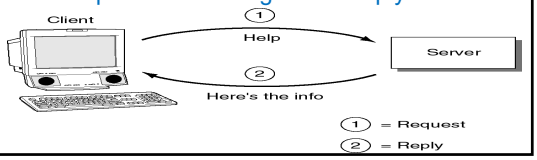
1.8.3.2 宏内核架构(13)

- 层级
  - 广泛应用于内核的资源管理中
    - 调度子系统中对进程优先级的分类
    - 内存分配器对不同内存的分类

86

1.8.3.3 微内核架构(1)

- 微内核结构
  - 操作系统从内核转移到“用户”空间
  - 以客户/服务器模式为基础
    - Server: a process implementing a certain service.
    - Client: uses the service by sending a request and waiting for the reply



1.8.3.3 微内核架构(2)

- 微内核的基本功能
  - 进程管理
    - 进程作为资源分配的基本单位
    - 线程作为独立运行和调度的基本单位
    - 进程与线程间的同步
  - 存储器管理
    - 为进程分配必要的运行空间
    - 提供虚拟存储器管理
    - 处理缺页中断

88

1.8.3.3 微内核架构(3)

- 进程通信管理
  - 用户模块之间通过传递消息进行通信
- I/O设备管理
  - 设备驱动程序
- 微内核结构的实现方式
  - 将最基本、最本质的OS功能保留在核中
  - 用户空间包含所有OS服务进程和用户应用进程
  - 每一OS功能均以单独的服务器进程形式存在

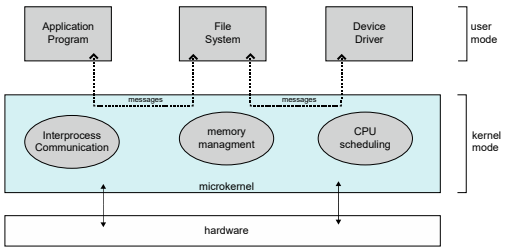
89

1.8.3.3 微内核架构(4)

- 进程以发送消息的方式通过微内核向服务器进程提出服务请求
- 服务器进程处理完请求服务后，以发送消息的形式通过微内核将结果返回客户进程
- 优点
  - 策略与机制进一步分离
    - 易于扩展
    - 易于移值
  - 服务与服务之间完全隔离
    - 更可靠
    - 更安全
- 缺点：性能

90

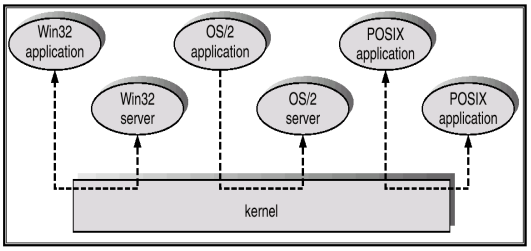
1.8.3.3 微内核架构(5)



微内核结构

91

1.8.3.3 微内核架构(6)



Windows NT结构

92

1.8.3.3 微内核架构(7)

- 微内核的发展
  - 第1代：Mach
    - RickRashid，Rochester+CMU
    - 对进程间通信(IPC)的设计过于通用，使得其性能弱于同时期的宏内核
  - 第2代：L4
    - Jochen Liedtke，德国国家信息技术研究中心
    - 极大提升了IPC的性能
    - 微内核最小化：一个操作系统内核的功能只有在其放在内核态以外会影响整个系统功能时，才能被放置在内核态

93

1.8.3.3 微内核架构(8)

- 第3代：增强安全性
  - Gernot Heiser
  - 将能力(Capability)机制引入微内核操作系统
    - 通过Capability进行IPC的权限判断
  - 代表系统
    - seL4：第一个完成形式化验证的内核
    - Google Fuchsia
    - MINIX：教学用微内核

94

1.8.3.4 外核架构(1)

- 引入
  - 操作系统内核对硬件资源进行了抽象
    - 过度抽象会带来性能损失
    - 通用抽象对具体应用往往不是最优选择
- 解决方案
  - 不提供硬件抽象
    - “只要内核提供抽象，就不能实现性能最大化”
    - 只有应用才知道最适合的抽象
  - 不管理资源，只管理应用
    - 保证应用之间的隔离
    - 负责应用与资源的绑定

95

1.8.3.4 外核架构(2)

- ExoKernel
  - 1995年，MIT的Dawson Engler和Frans Kaashoek等提出
  - 提出库操作系统(LibOS)概念
    - 将对硬件的抽象封装到LibOS中，以库形式提供，与应用直接链接
    - 按照应用领域特点高度定制化，不同应用动态组装成最适合该领域的LibOS，获得更高性能
    - 操作系统内核很小，多个LibOS之间具有强隔离性，提高安全性与可靠性

96

1.8.3.4 外核架构(3)

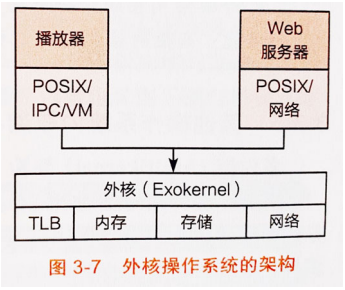


图 3-7 外核操作系统的架构

97

1.8.3.4 外核架构(4)

- 优点
  - OS无抽象，能够在理论上提供最优性能
  - 应用对计算有更精确的实时等控制
  - LibOS在用户态更易调试
- 缺点
  - 对计算资源的利用效率主要由应用决定
  - 定制化过多，维护难度增加
  - 缺乏跨场景的通用性，应用生态差

98

1.8.3.4 外核架构(5)

- 应用
  - 一些功能受限、对操作系统接口要求不高但对性能和时延特别敏感的嵌入式场景，可以将数据面与控制面分离
  - Unikernel(单内核)
    - 虚拟化环境下的LibOS
      - 每个虚拟机只使用内核态，内核态中只允许一个应用+LibOS，通过虚拟化层实现不同实例间的隔离
    - 适合容器等新的应用场景
      - 每个容器就是一个虚拟机，运行定制的LibOS以提高性能

99

1.8.3.5 多内核架构(1)

- 背景
  - 硬件结构呈现异构众核特点
- 多内核架构
  - 苏黎世联邦理工学院、微软研究院、雷恩高等师范学校等联合提出的一种实验型的操作系统内核架构
  - 基本想法
    - 将一个众核系统看成一个由多个独立处理器核通过网络互联而成的分布式系统
    - 假设硬件处理器提供全局共享内存语义
    - 为不同处理器核交互提供了一层基于进程间通信的抽象

100

1.8.3.5 多内核架构(2)

- 在每个CPU核上运行一个独立的操作系统节点，节点间的交互由操作系统节点之上的进程间通信来完成

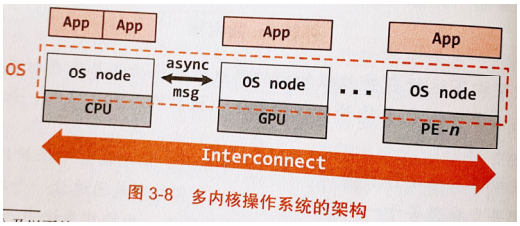


图 3-8 多内核操作系统的架构

101

1.8.3.5 多内核架构(3)

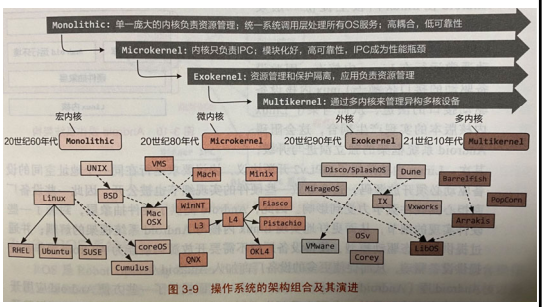


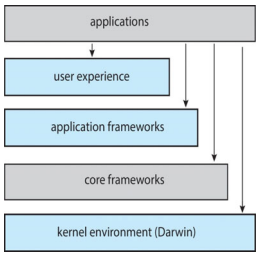
图 3-9 操作系统的架构组合及其演进

102

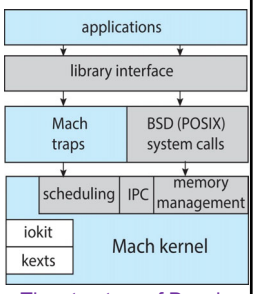
1.8.3.6 混合内核架构(1)

- 宏内核+微内核
  - 将需要性能的模块重新放回内核态
    - Windows NT=
      微内核+内核态的系统服务+系统框架
    - mscOS / iOS =
      Mach微内核+BSD 4.3+系统框架

1.8.3.6 混合内核架构(2)



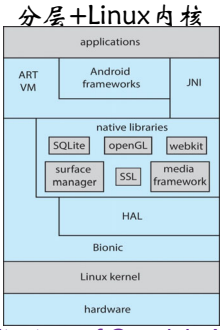
Architecture of Apple's macOS and iOS operating systems.



The structure of Darwin.

1.8.4 操作系统框架

- Android：分层+Linux内核



Architecture of Google's Android.

1.9 计算机系统体系结构(1)

- 核（core）
  - The core is the component that executes instructions and registers for storing data locally.
- 单处理器系统
  - If there is only one general-purpose CPU with a single processing core, then the system is a single-processor system.
- 多处理器系统
  - Traditionally, such systems have two (or more) processors, each with a single-core CPU.
  - The processors share the computer bus and sometimes the clock, memory, and peripheral devices.

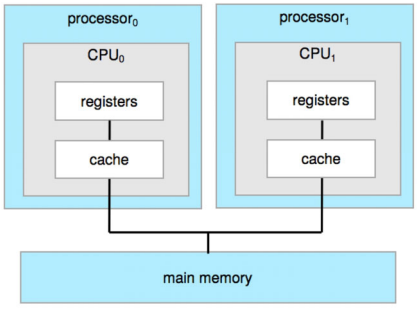
106

1.9 计算机系统体系结构(2)

- 类型
  - 非对称多处理器(Asymmetric Multiprocessing)
    - each processor is assigned a special task
  - 对称多处理器(Symmetric MultiProcessing , SMP)
    - each processor performs all tasks
    - all processors share physical memory over the system bus
- The definition of multiprocessor has evolved over time and now includes multicore systems, in which multiple computing cores reside on a single chip.

107

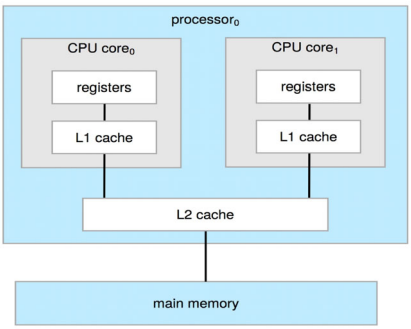
1.9 计算机系统体系结构(3)



Symmetric multiprocessing architecture.

108

1.9 计算机系统体系结构(4)



A dual-core design with two cores on the same chip.

1.9 计算机系统体系结构(5)

- 计算机系统组件定义
  - CPU
    - The hardware that executes instructions.
  - Processor
    - A physical chip that contains one or more CPUs.
  - Core
    - The basic computation unit of the CPU.
  - Multicore
    - Including multiple computing cores on the same CPU.
  - Multiprocessor
    - Including multiple processors.

本章要点

- 操作系统的定义
- 操作系统的特性和基本功能
- 操作系统的形成与发展
- 多道程序设计的概念和特点
- 操作系统的分类：批处理系统、分时系统、实时系统
- 操作系统的性能指标：资源利用率、系统吞吐量、作业（平均）周转时间
- 操作系统提供的服务
- 操作系统的两种运行方式
- 操作系统的结构