

# 学习操作系统的目的

- 理解计算机如何工作

- Philosophical issue for any CS degree holder
- A good computer scientist should understand what happens inside the system when one types a command

- 如何构建健壮的计算机程序

- OSes like Linux have many users and work on a wide range of hardware
- Deal with subtle issues: concurrency, consistency, etc

# 课程的基本要求

- 理解操作系统的发展、功能、地位和特点，并建立起以操作系统为中心的计算机系统的系统级认识和全局性把握。
- 理解操作系统的基本概念、原理、设计方法。
- 掌握操作系统中进程管理和调度、内存管理、文件管理等程序执行的关键技术。
- 通过了解操作系统 API，深入理解操作系统的工作原理；掌握并发程序设计的基本技术。
- 掌握1至2个主流操作系统，熟练使用并具有剖析、修改、替换和扩充系统模块的初步能力。

# 课程地位

- 操作系统是计算机系统的核心和灵魂，是硬件的首次扩充，又是最重要的系统软件，该课程具有承上启下的重要作用。
  - 对前导课程进行总结和提高
    - **计算系统基础、计算机组成原理、数据结构、软件工程**
  - 为后继专业课程的学习打下良好基础
    - **嵌入式系统、服务计算、分布式计算、网格计算、云计算等**



**“Learn OS concepts by coding them!”**

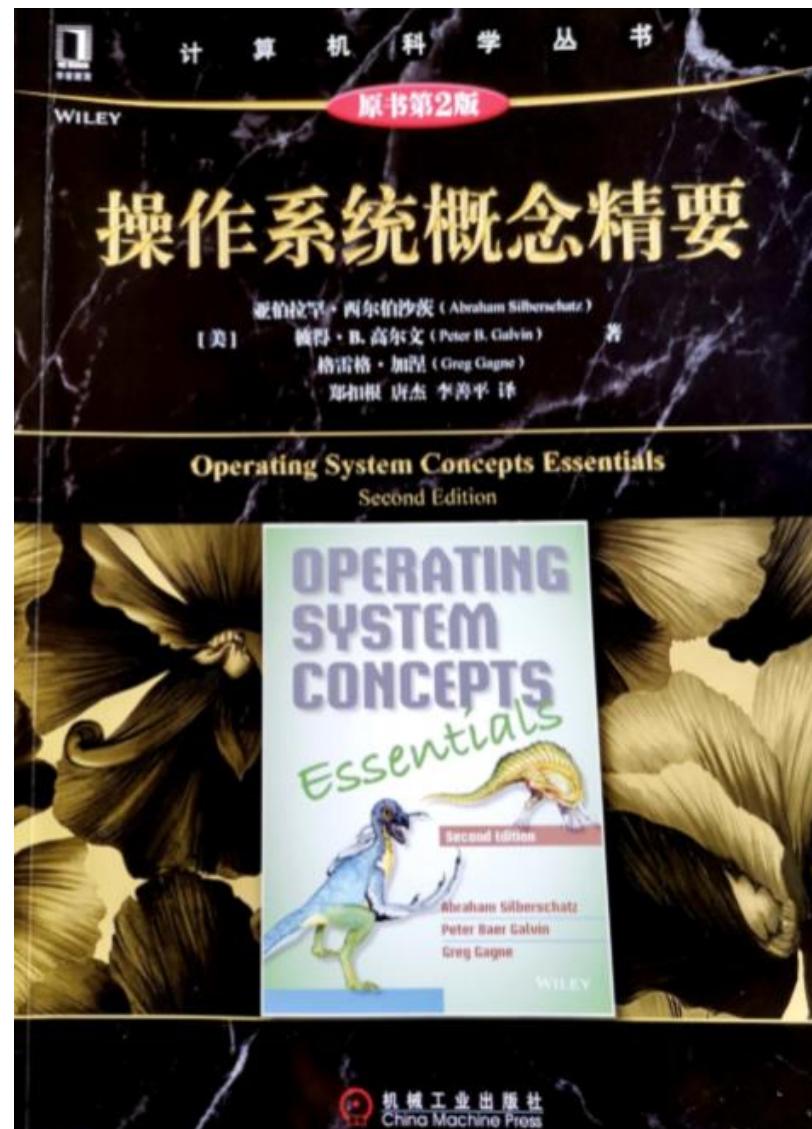
OS中有很多并不深刻的概念(如分时、实时等)，记住这些概念没有意义，实现它们才是最重要的。

OS是最复杂、最基础的软件系统，coding them是计算机专业的学生的责任!



# 教材

Abraham Silberschatz著. 郑扣根等译. 操作系统概念精要(第2版).  
机械工业出版社, 2018

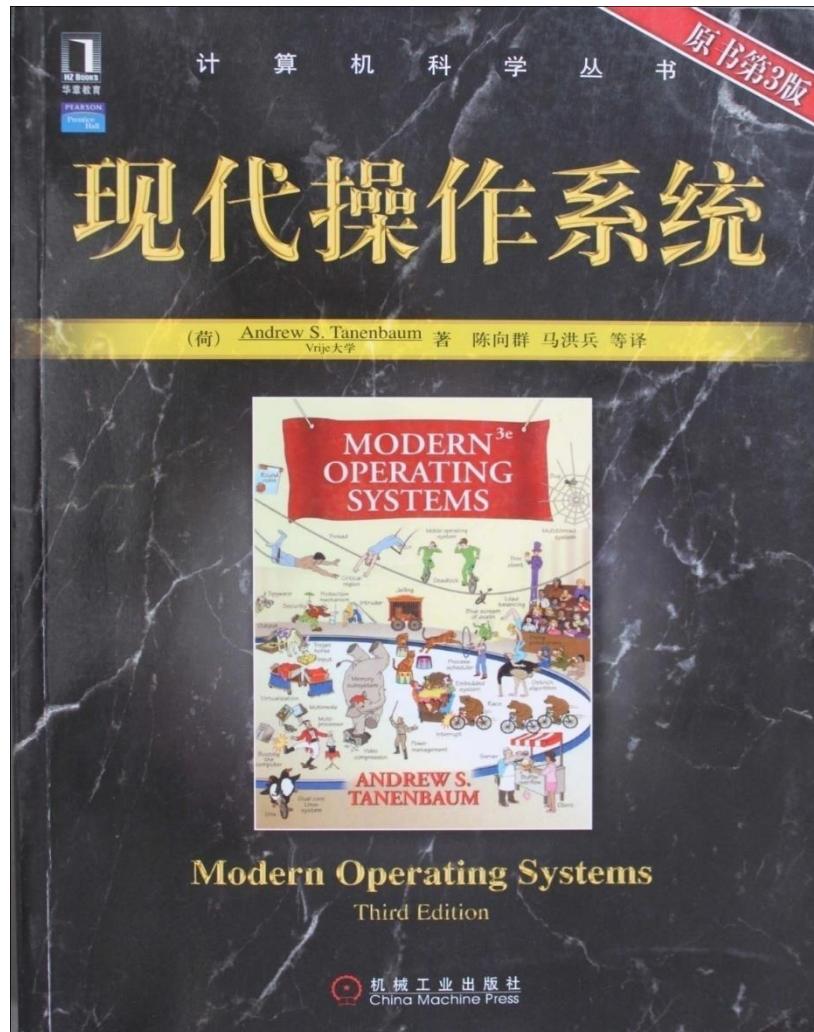


# 实验教材

刘宁等 编著. 计算机操作系统实验与课程设计指导书.  
大连海事大学出版社, 2017年

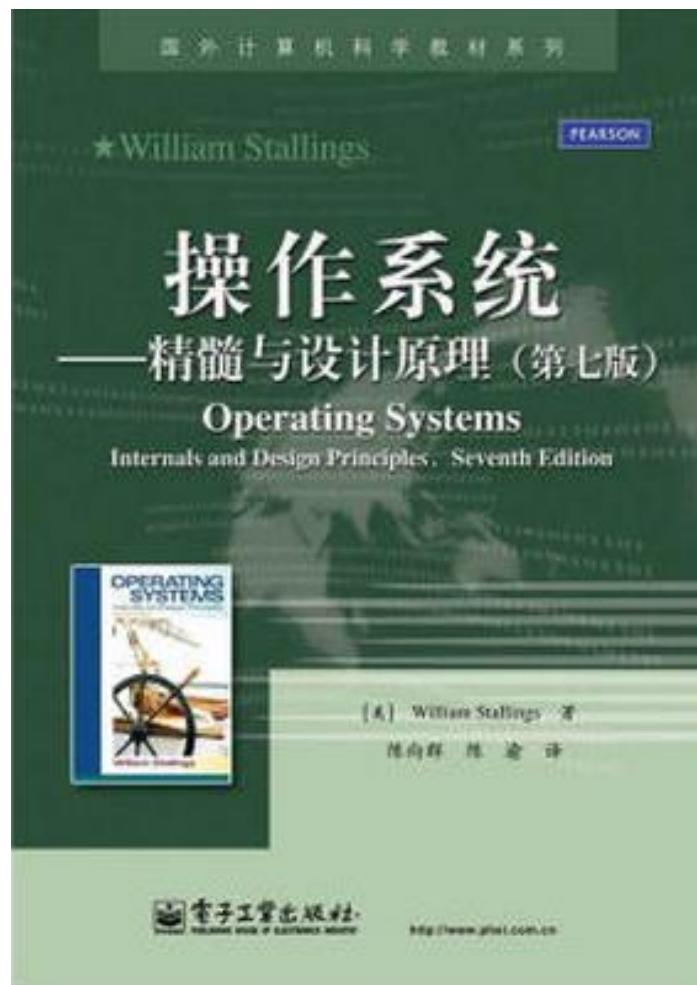
# 参考书 1

Andrew S. Tanenbaum等著. 陈向群等译. 现代操作系统(第4版).  
机械工业出版社, 2017



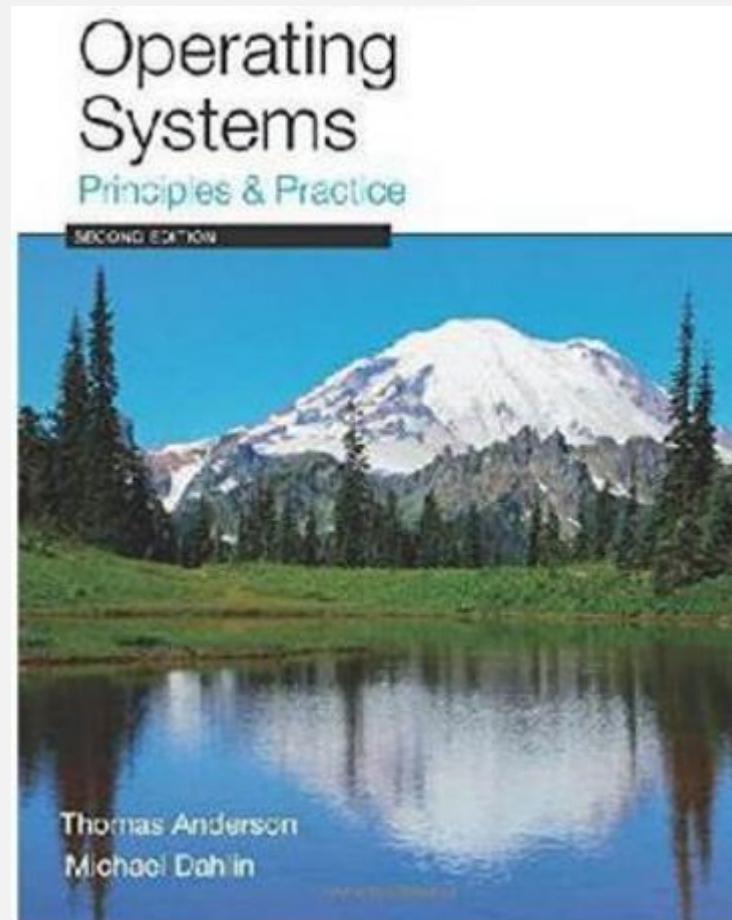
## 参考书 2

William Stallings 著. 陈向群等译. 操作系统：精髓与设计原理.  
电子工业出版社, 2017



# 参考书 3

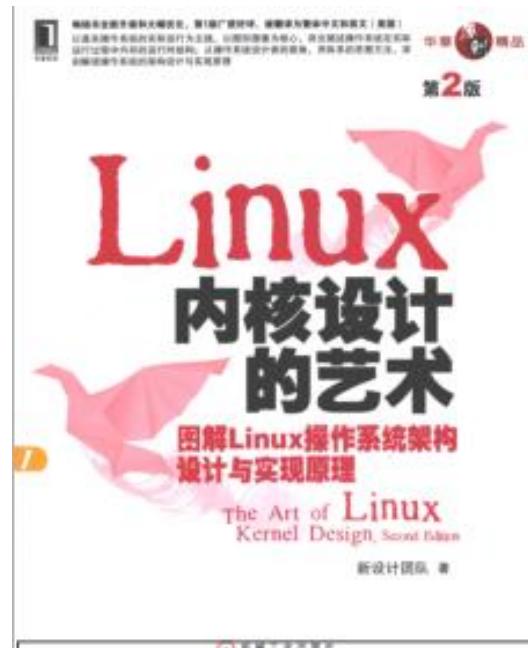
Thomas Anderson. Operating Systems: Principles and Practice.  
Recursive Books



# 参考书 4

新设计团队. LINUX内核设计的艺术:图解LINUX操作系统架构设计与实现原理 (第2版). 机械工业出版社

以图形、图像为核心，突出描述操作系统在实际运行过程中内存的运行时结构；强调学生站在操作系统设计者的视角，用体系的思想方法，整体把握操作系统的行为、作用、目的和意义。



# 第1章 导论

1.1 操作系统做什么

1.2 计算机硬件系统的组成

1.3 计算机系统的体系结构

1.4 操作系统的执行

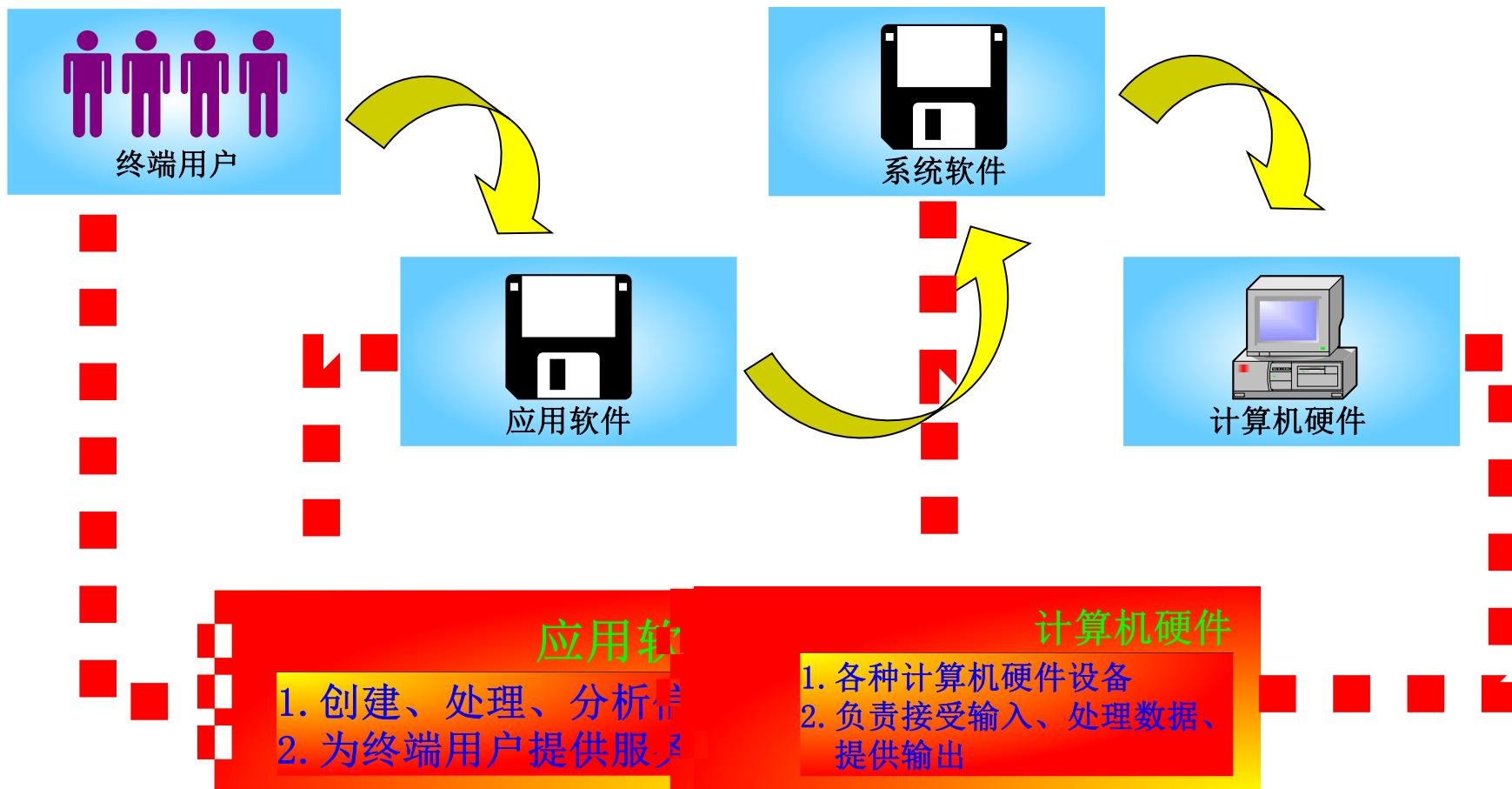
1.5 操作系统的功能

1.6 操作系统的发展

1.7 操作系统对一个程序的处理过程

# 1.1 操作系统做什么

## 计算机系统



# 计算机系统

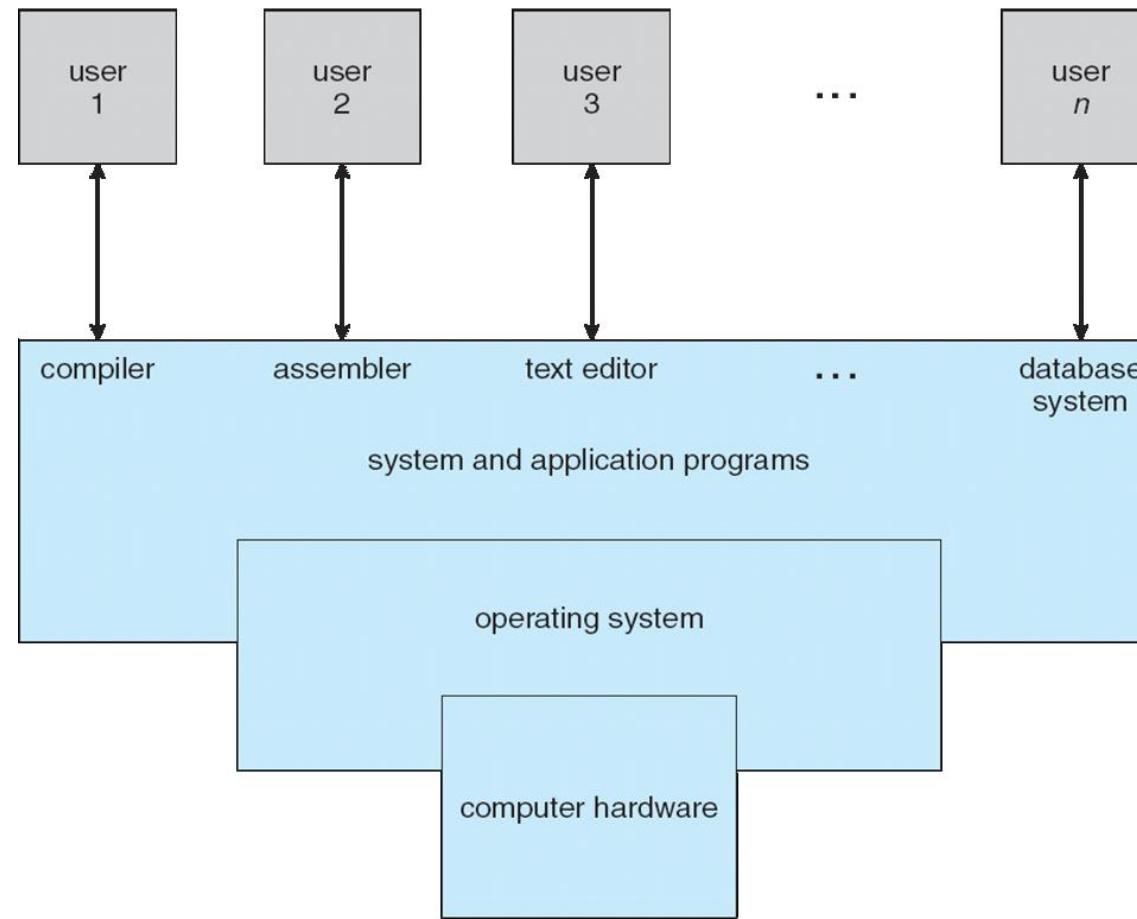
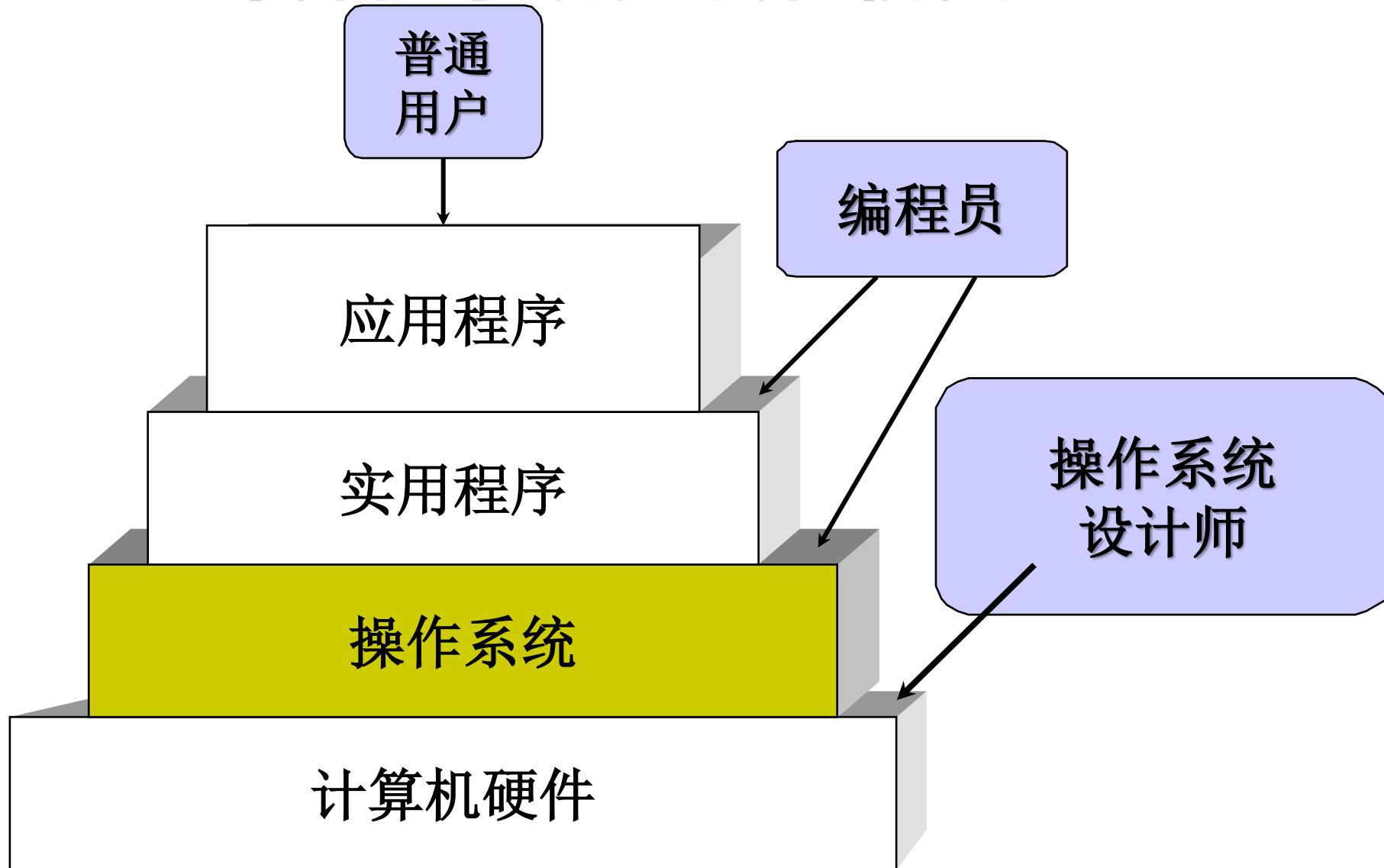


图 1-1 计算机系统组件的抽象视图

# 计算机系统分层和视点



# 操作系统的定义

没用公认的标准定义

- 计算机系统中的一个系统软件，一些程序模块的集合
- 管理和控制计算机系统中的软件和硬件资源
- 合理地组织计算机工作流程
- 在计算机与其用户之间起到接口的作用。



# 1.2 计算机硬件系统的组成

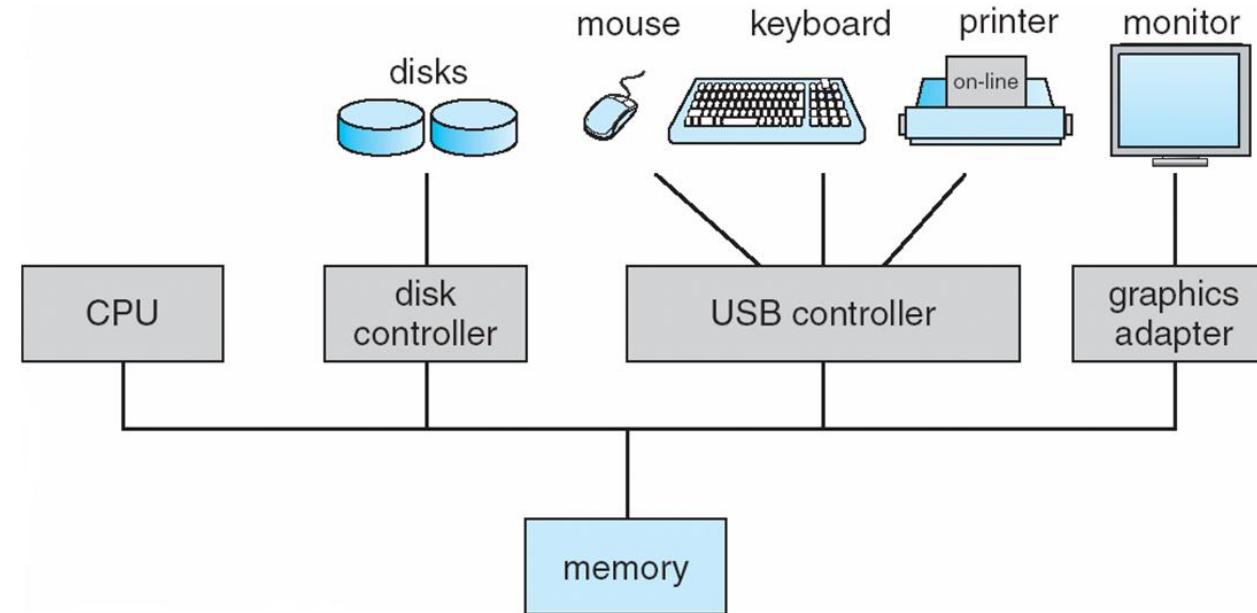


图 1-2 计算机系统

# 计算机系统的运行 (1)

- 一个或多个CUP和若干设备控制器、通过公用总线相连
  - 总线提供了对共享内存的访问
- CUP和设备并发执行、竞争访问内存
  - 每种设备控制器负责一种特定的设备类型
  - 每个设备控制器有一个local缓冲寄存器
  - CPU 在主存和缓冲寄存器之间移动数据
- 设备通过发出中断通知CPU已完成操作



# 计算机系统的运行 (2)

## ■ 开机时运行引导程序

- 位于ROM或者EEPROM(电可擦可编程只读内存)
- 初始化系统各个组件，从CPU寄存器、设备控制器到内存内容
- 定位外存硬盘上的操作系统内核并且加载到内存

## ■ 内核加载到内存开始运行

- 设置和初始化各种数据结构与表格、建立各种控制系统内核运行的系统进程和系统后台进程
- 等待事件发生

## ■ 事件发生通过硬件和软件中断来通知

- 硬件中断的触发：硬件随时通过系统总线发送信号到CPU
- 软件中断的触发：程序中发出系统调用



# 中断机制的功能

- 将控制转移给中断服务程序

- 通过中断向量，它包含了所有中断服务程序的入口地址

- 保存中断指令的地址

- 在系统堆栈上
  - 处理完中断，保存的返回地址会加载到程序计数器，被中断的程序继续执行。

- 操作系统是中断驱动的



# 存储结构

## 存储定义与符号

### ■ bit (位、比特)

- 存储的基本单位，所有其它计算机存储都是由位组合而成
- 位数足够，计算机能够表示各种信息：数字、字母、图像、视频、音频、文档和程序。

### ■ Byte (字节)

- 8位、大多数计算机的常用最小存储。

### ■ Word (字)

- 给定计算机架构的常用存储单位，每个字由多个字节组成。
- 如：一个具有64位寄存器和64位内存寻址的计算机通常采用64位的字（8字节）。
- 计算机许多操作通常以字为单位。



# 存储定义与符号

$2^{10}$  bytes = **1KB**

$2^{20}$  bytes = **1MB**

$2^{30}$  bytes = **1GB**

$2^{40}$  bytes = **1TB**

$2^{50}$  bytes = **1PB**



# 存储结构

## ■ 内存高速缓存

- 计算机系统结构的一部分、由硬件实现，对操作系统不可见。

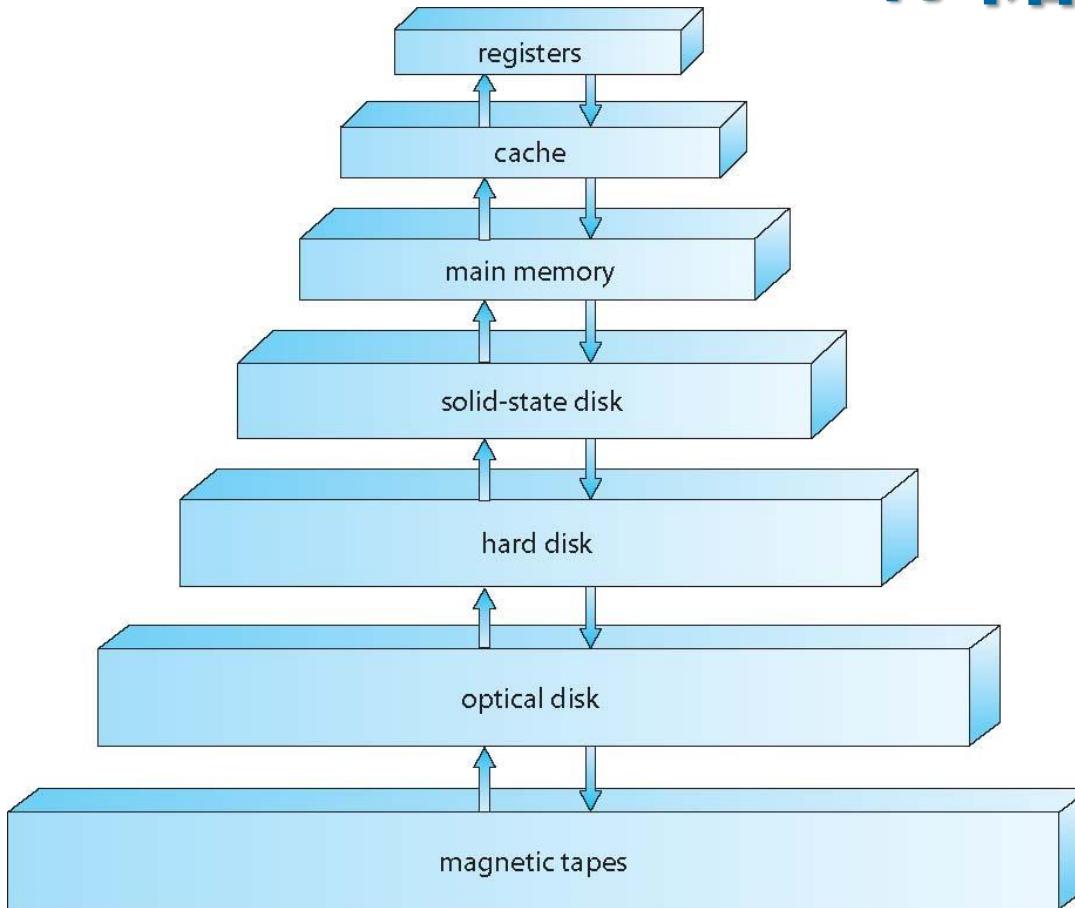
## ■ 主存(main memory)(Random Access Memory)

- CPU能直接访问
- 易失的

## ■ 二级存储：主存的扩充、大容量不易失

- 硬盘、磁盘
- 固态硬盘：低功耗、无噪音、体积小、发热量小，读写速度远高于传统硬盘

# 存储设备的层次



## 存储器的三个特性

- 容量、存取速度和价格
- 存取速度越快，每个“位”的价格越高。
- 容量越大，每个“位”的价格越低。
- 容量越大，存取速度越慢。

图 1-4 存储设备的层次

**层次结构成功的关键：低层访问频率递减**

# 1.3 计算机系统的体系结构

## ■ 单处理器系统

## ■ 多处理器系统

■ 优点：增加吞吐量、规模经济、增加可靠性

### ■ 非对称多处理器

■ 主处理器：一个、只运行OS、管理整个系统的资源，为从处理器分配任务；

■ 从处理器：可有多个，执行应用程序或I/O处理；

■ 不同性质任务的负载不均，可靠性不够高。

### ■ 对称多处理器（Symmetric MultiProcessing）

■ 每个处理器都参与完成操作系统的所有任务，没有主从关系；

■ 任务负载较为平均，性能调节容易；

■ 几乎所有现代操作系统支持，如Windows Mac OSX和Linux，



# 集群系统 (Clustered Systems)

## ■ 由两个或多个独立的系统 (或节点) 组成

- 每个节点可为单处理器系统或多核系统
- 集群计算机共享存储

## ■ 高可用性

- 集群中的一个或多个系统出错，仍可继续提供服务

## ■ 对称的

- 两个或多个主机都运行应用程序、并互相监视
- 充分使用现有硬件、当多个应用程序可供执行时，更有效

## ■ 非对称的

- 一台处于热备份模式，另一台运行应用程序

## ■ 应用

- 高性能计算(High-Performance Computing, HPC): 并行计算 (parallelization)，将一个程序分成多个部分，并行运行在各个核上
- 分布锁管理器(Distributed Lock Manager, DLM): 多个主机访问共享存储的同一数据

# 1.4 操作系统的执行

现代操作系统是中断驱动的(Interrupt driven)

- 硬件中断：设备发出的
- 软中断(陷阱或异常exception or trap)
  - 软件出错(如除数为0、无效的存储访问)、进程试图修改别的进程或OS等
  - 进程发出系统调用，需要操作系统服务

OS和多个进程共享计算机系统的软硬件资源，  
为确保OS的正确运行，区分OS代码和用户代码的运行。

多数系统将处理器状态划分为管态和目态。



# 双重模式与多重模式的执行

根据运行程序对资源和机器指令的使用权限划分

## ■ 管态（特权态、系统态、内核态）

- 操作系统程序运行时的状态，较高的特权级别
- 可以执行所有的指令（包括特权指令）、使用所有的资源，并具有改变处理器状态的能力。

## ■ 目态（普通态、普态、用户态）

- 用户程序运行时的状态，较低的特权级别；
- 禁止使用特权指令，不能直接使用系统资源与改变CPU状态，只能访问用户程序所在的存储空间。

# 受保护的指令(特权指令)

- 访问某些硬件资源的指令；
  - 对I/O设备直接访问的指令，如磁盘、打印机等；
- 对内存管理状态进行操作的指令；
- 某些特殊的状态位的设置指令；
- 停机指令。

只有操作系统才有权使用



# x86系列处理器的执行模式

386、486、Pentium系列支持4个处理器特权级别

■从R0到R3特权能力依次降低

■R0：双状态系统的管态，运行操作系统核心代码

■R3：目态

■R1：运行关键设备驱动程序和I/O处理例程

■R2：运行其它受保护共享代码，如语言系统运行环境

多数Unix、Linux以及Windows系列  
只用R0和R3两个特权级别



CPU如何来判断当前运行的程序是  
系统程序还是用户程序呢？

## 程序状态字PSW(Program Status Word )

- 状态码：管态还是目态，决定是否可以使用特权指令或拥有其它的特殊权力；
- 条件码：指令执行后的结果特征；
- 中断屏蔽码：是否允许中断

# CPU工作状态之间如何转换？

- 管态 → 目态      设置PSW
- 目态 → 管态      用户程序无法直接修改程序状态字；  
                        通过系统调用



# 系统调用

操作系统内核函数

用户程序通过访管指令，请求操作系统提供某种服务。

- 当CPU执行访管指令时，引起访管中断；
- 处理器保存中断点的程序执行上下文环境（PSW、PC和其它寄存器），**CPU切换到管态**(硬件自动完成)。
- 中断处理程序开始工作，调用相应的系统服务；
- 结束后，恢复被中断程序的上下文环境，**CPU恢复为目态**，回到中断点继续执行。

# 用户模式到内核模式的转换

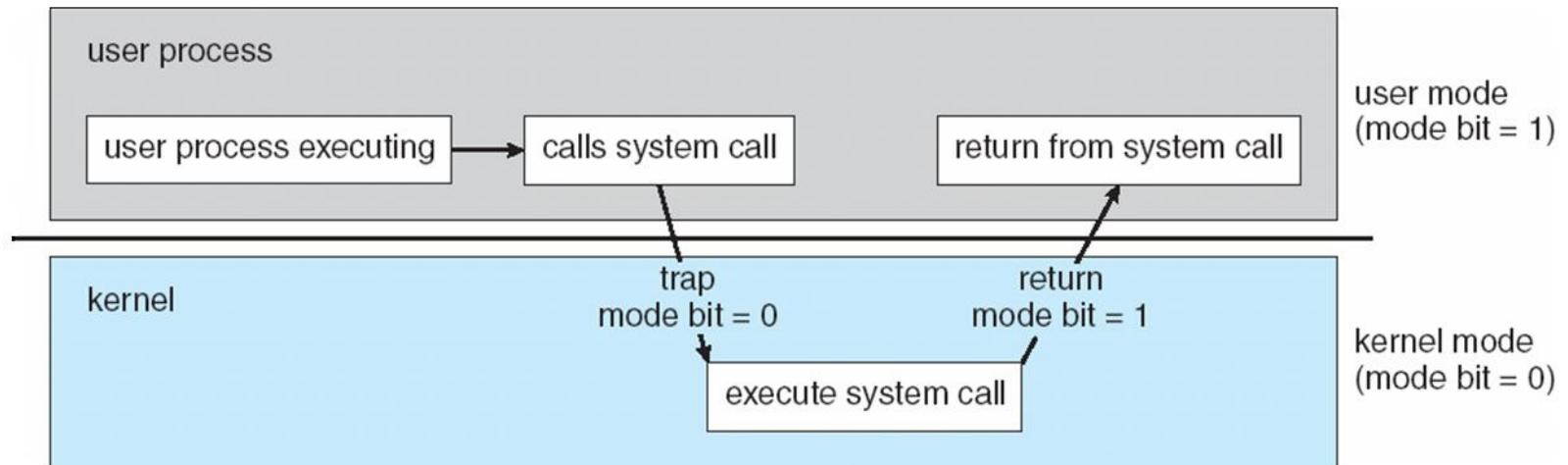
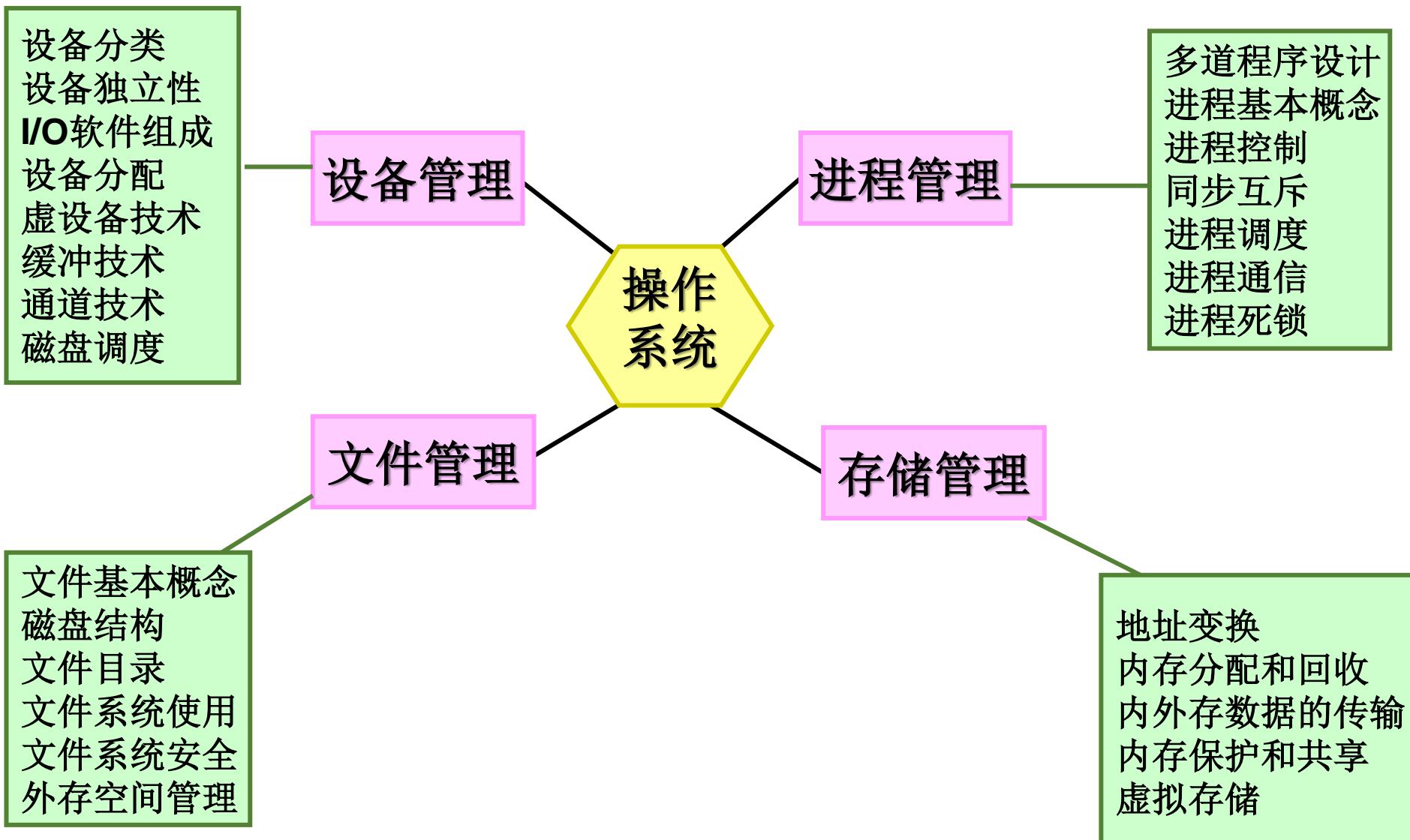


图 1-10 用户模式到内核模式的转换

# 1.5 操作系统的功能



# 1.6 操作系统的发展

## ■发展动力：“需求推动发展”

- 设备角度：资源利用率和系统性能
- 用户角度：方便用户
- 技术角度：物理器件发展

## 操作系统的發展历史

## 操作系统的现状



# 操作系统的发展历史

- (1945–55) 真空管
- (1955–65) 晶体管和批处理系统
- (1965–1980) 集成电路和多道程序设计
- (1980–1995) PC
- (1995–Present) 后PC



# 真空管

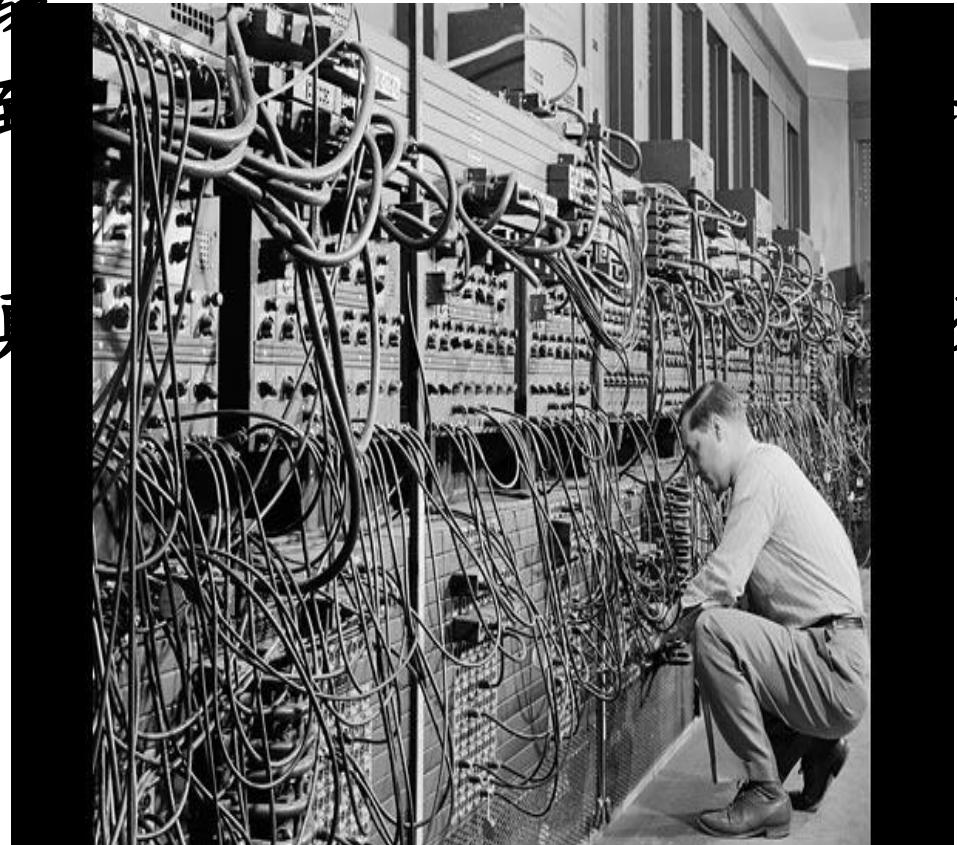
## ENIAC 计算机

- 运算速度：1000次/秒；
- 数万个真空管，占地100平方米。
- 没有程序设计语言 没有操作系统

程  
机  
里  
期  
算  
自



然后到  
几万个真



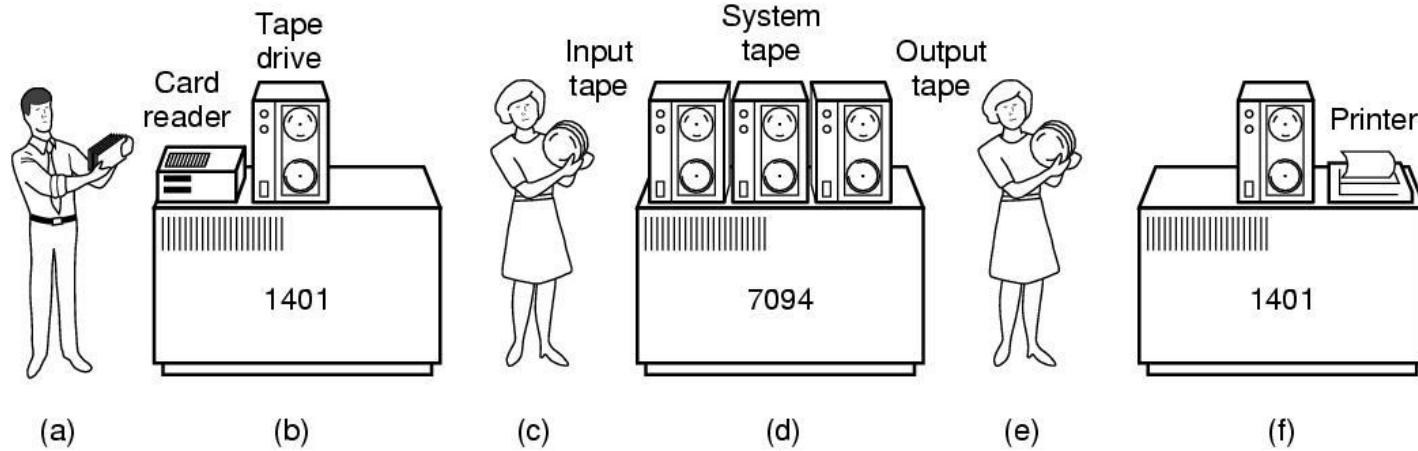
计算  
以计

# 晶体管和批处理系统(1)

- 晶体管计算机，操作系统的雏形--监督程序。
- 把若干个作业合成一批，调入计算机执行，完成后再调入下一批作业。
- 计算机的应用从数值计算扩大到数据处理、工业过程控制等领域。



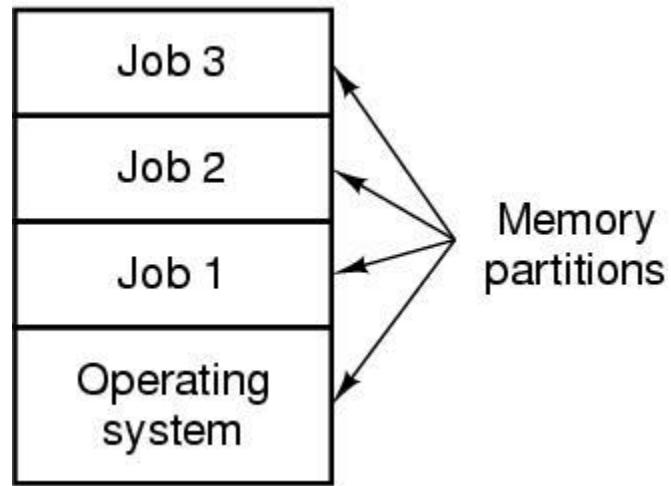
# 晶体管和批处理系统(2)



## An early batch system:

- (a) Programmers bring cards to 1401.
- (b) 1401 reads batch of jobs onto tape
- (c) Operator carries input tape to 7094.
- (d) 7094 does computing.
- (e) Operator carries output tape to 1401.
- (f) 1401 prints output.

# 集成电路和多道程序设计



内存中存放三个作业的多道程序系统



# 多道程序设计技术

**在内存中放多道程序，  
在管理程序的控制下相互穿插地运行。**

形成的动力：提高资源利用率和系统吞吐量。

硬件基础：60年代通道技术、中断技术

**操作系统发展史上革命性变革。**

# 多道程序系统的特点

## ■ 多道

- 计算机内存中同时存放几道相互独立的程序。

## ■ 宏观上并行

- 同时有多道程序在内存运行
- 某一时间段上，各道程序不同程度地向前推进。

## ■ 微观上串行

- 任一时刻最多只有一道作业占用CPU（单CPU），多道程序交替使用CPU。



# 多道程序系统的优缺点

## ■优点

- 资源利用率高
- 系统吞吐量大
  - 系统吞吐量：系统在单位时间完成的总工作量。

## ■缺点

- 平均周转时间长
  - 作业的周转时间：从作业进入系统开始，直至其完成并退出系统为止所经历的时间。
- 无交互能力

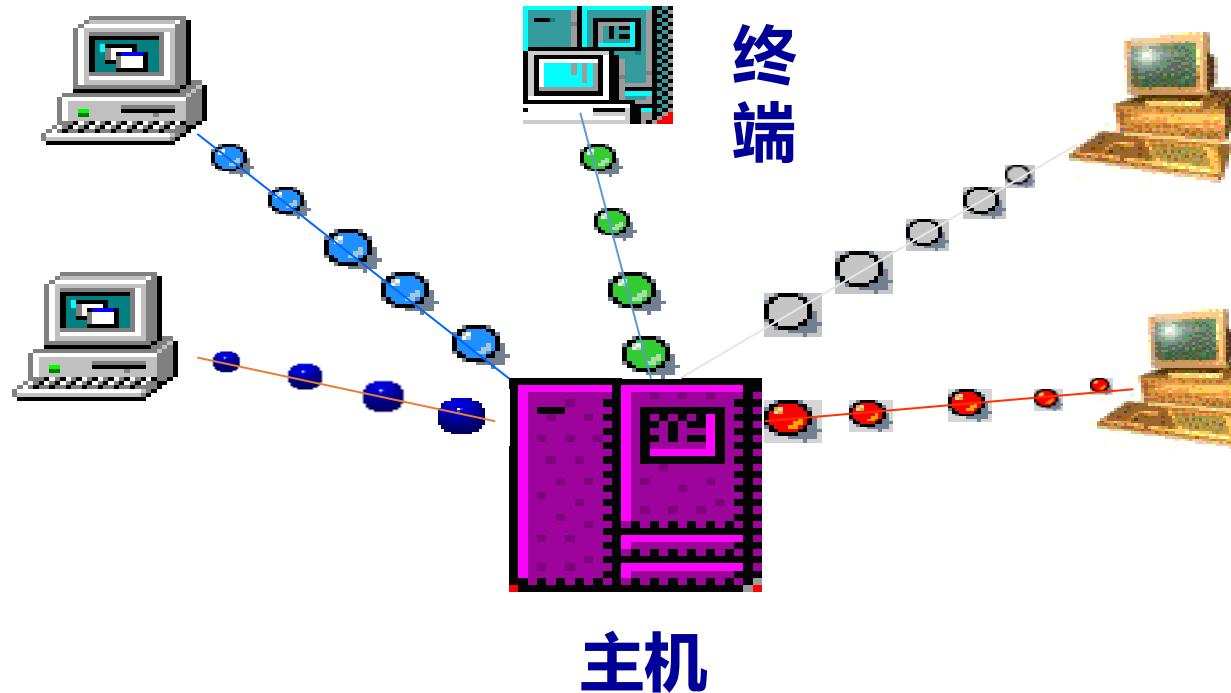


# 分时操作系统

## ■ 形成和发展的动力

- 用户的需要：交互、共享主机、方便上机。

在一台主机上连接多个带有显示器和键盘的终端，同时允许多个用户通过自己的键盘，以交互的方式使用计算机，共享主机中的资源。



# 分时系统的特征

## ■多路性

- 允许同一主机联接多台终端。

## ■独立性

- 每一用户独占一个终端。

## ■及时性

- 用户请求能及时响应。

## ■交互性

- 可进行广泛的人机对话。

# Personal Computers

## ■ DOS

- 单用户单任务操作系统

## ■ IBM OS/2

- 80286保护方式下工作的单用户多任务操作系统。
- 支持16个任务并发执行。

## ■ Windows

- 1985年11月20日，Windows 1.0正式上市
- 个人计算机上最广泛使用的操作系统。

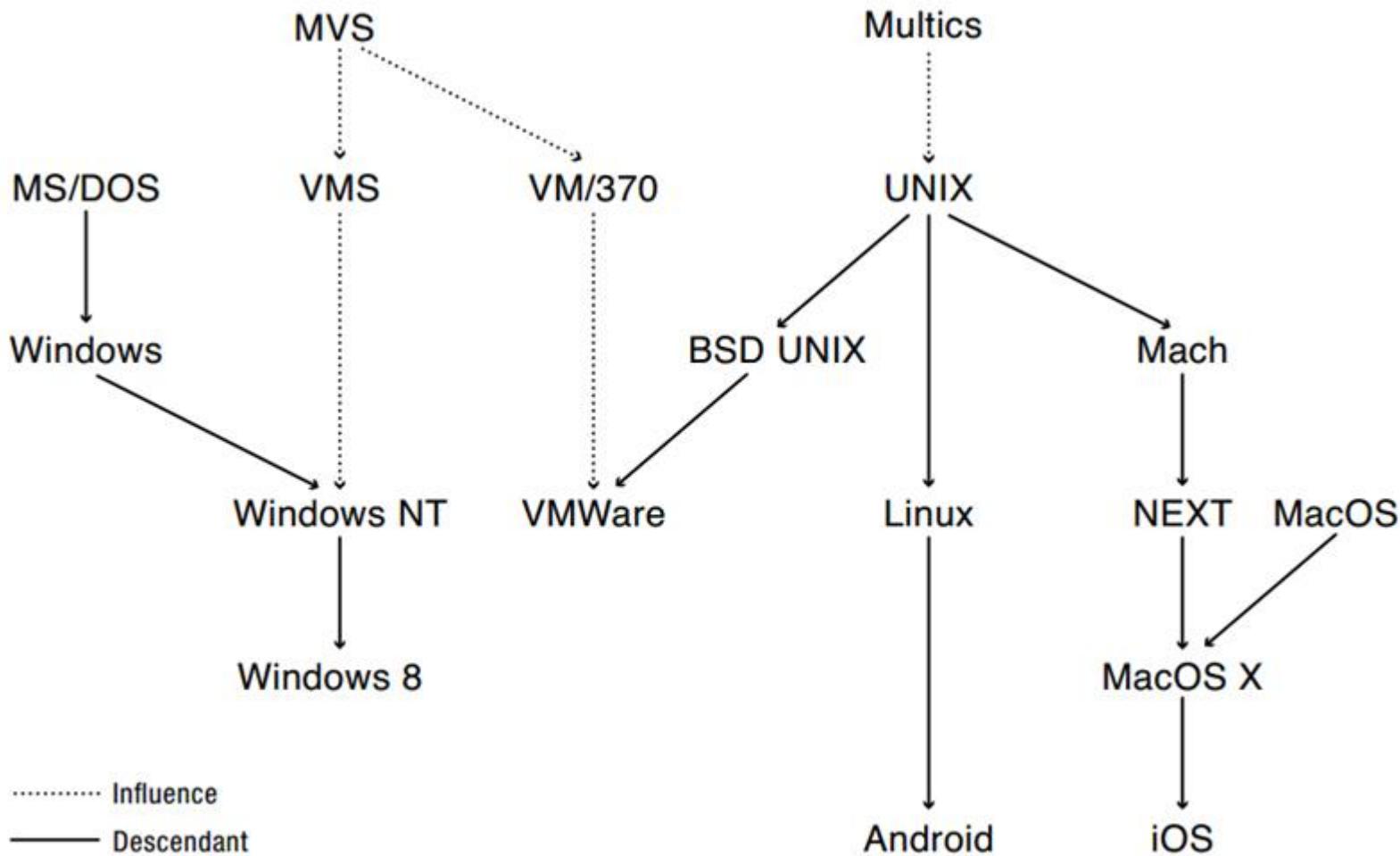


# 后PC

- 时代环境：移动计算、无线计算、嵌入式计算
- 特点：小型、移动、便捷、有限能力
  - 移动计算的新潮：智能手机、高性能笔记本电脑
  - 工业智能化的趋势：形形色色的嵌入式系统



# OS History



Genealogy of several modern operating systems

Thomas Anderson , Operating Systems: Principles and Practice



# 操作系统的现状

## ■ 桌面操作系统

- Windows、Linux、MacOS 等

## ■ 服务器操作系统

- Unix/类Unix系列
- Windows 等

## ■ 智能手机操作系统和平板电脑操作系统

### ■ Android

- 基于安卓的生态更庞大、繁荣、高频

- 微信、商务、游戏、生活、娱乐

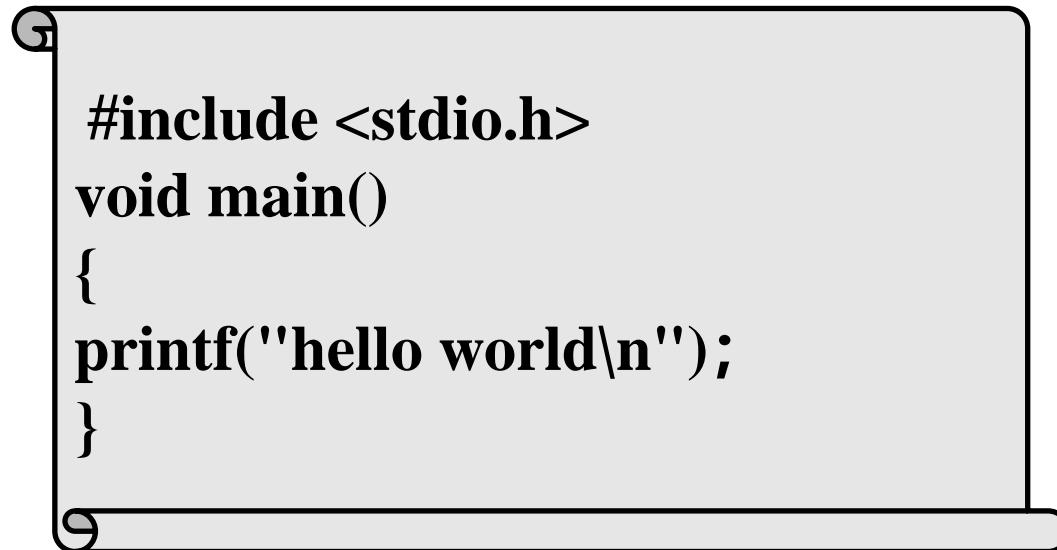
- 全球无数团队，基于安卓在源源不断地创意、开发新的应用

### ■ Apple iOS。



# 1.7 操作系统对一个程序的处理过程

## “hello world” 在Linux上运行



涉及操作系统的代码在1万行以上，  
除进程间通信外，几乎涉及操作系统的方方面面。

LINUX内核设计的艺术: 图解LINUX操作系统架构设计与实现原理

# 1. 用户输入命令，shell进程被唤醒，对命令解析。

- 用户敲击键盘后，键入的信息记录在终端设备文件（tty0）上。
- 产生键盘中断信号，系统对键盘中断信号进行处理。
- 中断服务程序开始执行后，唤醒shell进程，通过进程调度机制，由进程0切换到shell进程去执行。
- shell进程从tty0终端设备文件上读取用户键入的指令信息，解析该指令，并准备进行相应的处理。

## 2. 调用fork创建一个用户进程，对程序进行控制。

- 创建进程控制块 (task\_struct)
  - 进程状态、进程对应的文件
  - 进程调度信息
  - 内存管理信息
  - 文件管理信息



### 3. 加载hello world文件对应的程序

- 文件系统：解析文件路径、操作目录文件和目录项、操作i节点表等
- 内存管理：页面引用计数、页面三级管理机制（页目录表、页表、页面）、页面数据、地址映射机制、缺页中断机制。
- 文件和内存是所有进程可以共用的资源，它们之间还存在着更为复杂的管理关系。
  - 如：两个进程加载同一个hello world文件时，涉及要不要共享，如何共享，共享后页面的引用计数如何计算，读写属性如何确定等。



## 4. 程序开始执行，将“hello world”显示在屏幕上。

- **设备管理**：直接与显示器的底层交互。
- 确定操作合法，然后将字符串转换成像素
- 将像素写入存储映像区
- 视频硬件将像素表示转换成一组模拟信号控制显示器
- 你在屏幕上看到“hello world”



# 如果没有操作系统？

- 即使是在屏幕上显示一行“hello world”，都要写大量复杂的、具有操作系统所具备的功能的程序
- 甚至无法把程序加载到计算机中，更谈不上得到运行结果了。



# 操作系统为应用程序的运行做了些什么？

- 提供了对外设的支持
- 支持多个程序同时运行
  - 对运行的多个程序进行有效的组织、管理和协调，防止某个程序独占CPU、内存、外设等资源。
  - 防止正在运行的程序之间相互读写和相互覆盖，确保所有程序正确运行。
  - 操作系统不能被应用程序直接读写，更不能被应用程序覆盖。



# 本章要求

- **掌握**
  - 操作系统概念
  - 操作系统的功能
- **了解**
  - 操作系统运行的两种模式
  - 操作系统分类和各类操作系统的特点