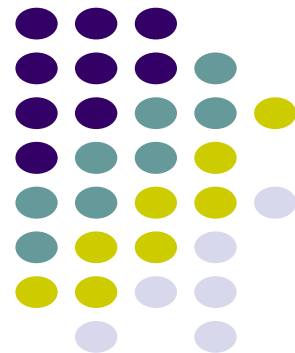
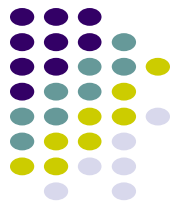


# 操作系统

## 第1章 操作系统引论



# 简介



- 课程简介：操作系统是一门涉及较多硬件的系统软件课程，在计算机软硬件课程的设置上起着承上启下的作用。既涉及硬件资源管理又涉及软件算法，是计算机课程的专业主干课。
- 课程特点：其特点是概念多、较抽象和涉及面广，整体实现思想和技术又比较难于理解。
- 学习方法：牢记概念 理解功能 联系实际 前后贯通
- 主要知识：操作系统的四大管理功能

**OS 是什么，做什么，如何做**



# 操作系统课程的重要性

- 掌握核心系统软件
- 掌握并发处理的思想方法
- 考研专业课的重要科目
- 为后继课程打好基础

# 考研专业课统考



- 分数比例

150分，180分钟

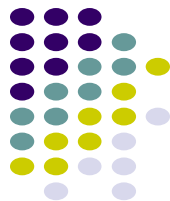
数据结构 45分；计算机组成原理 45 分

操作系统 35分；计算机网络 25分

- 试卷结构

单项选择题 80分（40小题×2分）

综合应用题 70分



## 主要参考书

- 《计算机操作系统》（第4版），汤小丹等 西安电子科技大学出版社
- 《计算机操作系统(第3版)》 汤小丹等 西安电子科技大学出版社
- 《操作系统教程——原理和实例分析》（第3版） 孟静 高教出版社
- 《操作系统考研指导》 清华大学出版社



# 课程内容

- 操作系统引论
- CPU管理
- 存储管理
- 设备管理
- 文件管理
- 操作

I/O系统  
I/O控制方式  
缓冲技术  
I/O软件组成  
设备独立性  
设备分配  
驱动程序  
虚设备技术  
通道技术  
磁盘调度

多道程序设计  
进程基本概念  
进程同步互斥  
进程间通信  
进程调度  
死锁

操作系统定义  
OS的作用  
OS特征  
OS的主要功能  
OS目标  
OS分类

处理机管理

基本概念

操作系统

设备管理

作业管理  
用户接口

用户接口  
作业基本概念  
批处理系统作业管理  
分时系统作业管理

文件管理

存储管理

文件基本概念  
文件的逻辑结构  
文件的物理结构  
文件目录  
外存空间管理  
文件共享与保护  
数据一致性

程序的装入与链接  
存储管理任务  
动态分区分配  
交换技术  
页式存储管理  
段式存储管理  
段页式  
虚拟存储技术

# 操作系统（OS）的定义

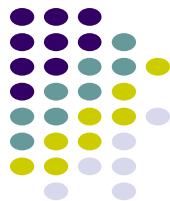
- 你用过哪些OS?
- OS能做什么?
  - ✓ 各种命令: `dir` `copy` `del` `format`
  - ✓ 启动、结束用户程序
  - ✓ 系统调用
  - ✓ UNIX 等提供多任务、多用户环境

NOKIA  
symbian



Mac OS





# 操作系统（OS）的定义

- OS不能做什么？
- OS是什么？

不做文档编辑

不做房屋设计

不是编译程序 ... ..

结论：OS不直接解决最

终具体应用问题，为你完

成所有“硬件相关、应用

无关”的工作，以给你方

便、效率、安全。

任务管理器

文件(F) 选项(O) 查看(V)

进程 性能 应用历史记录 启动 用户 详细信息 服务

名称	8% CPU	61% 内存	0% 磁盘	0% 网络	46% GPU
> Google Chrome (6)	0%	422.8 MB	0 MB/秒	0 Mbps	0%
> Windows 资源管理器	0.5%	260.2 MB	0 MB/秒	0 Mbps	0%
Broadcast DVR server	0.8%	195.7 MB	0 MB/秒	0 Mbps	20.2%
> 腾讯QQ (32 位) (2)	0.2%	138.9 MB	0.1 MB/秒	0 Mbps	0%
> WPS Presentation (32 位) (2)	0.2%	117.1 MB	0 MB/秒	0 Mbps	0%

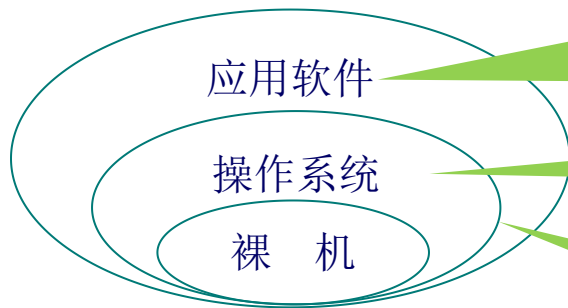
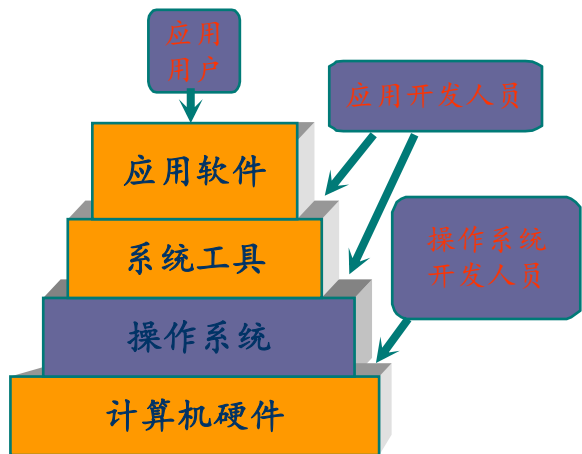
对硬件  
的管理

对软件  
的管理

OS是直接控制和管理计算机硬件、软件资源，合理地对各类作业进行调度，以方便用户使用的程序集合



# OS在计算机中的地位



操作系统功能示意图

应用程序：如QQ、office、英雄联盟、绝地求生、迅雷...

如：Windows 10

裸机：包含 CPU、内存、硬盘、主板等



# 操作系统的目标和作用

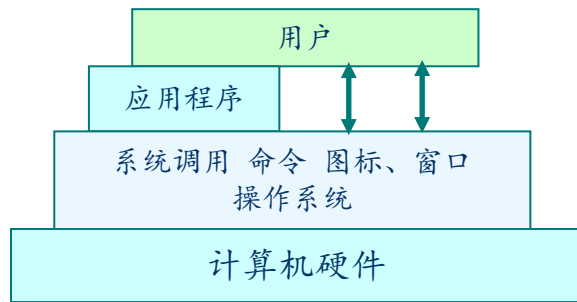
- 操作系统的目标与应用环境有关。例如在查询系统中所用的OS，希望能提供良好的**人—机交互性**；对于应用于工业控制、武器控制以及多媒体环境下的OS，要求其具有**实时性**；而对于微机上配置的OS，则更看重的是其使用的**方便性**。
- 方便性、有效性、可扩充性、开放性

# OS的作用



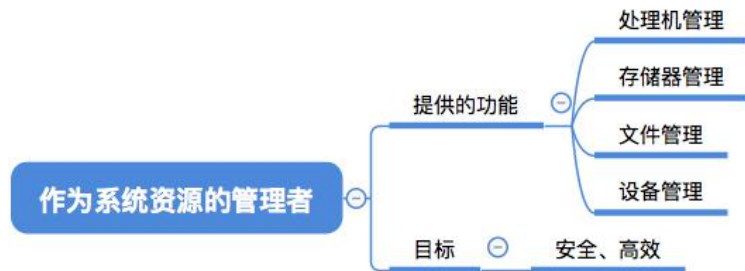
- **作为用户和计算机间的接口**

OS处于用户与计算机硬件系统之间  
用户通过OS来使用计算机系统



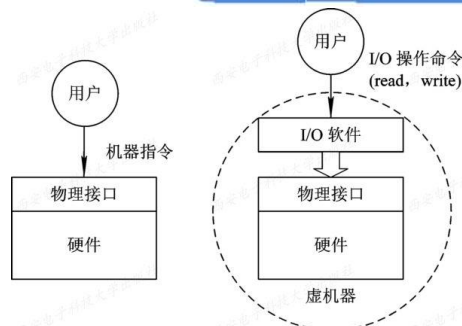
- **作为计算机系统资源的管理者**

处理机管理、存储器管理、  
设备管理、文件管理



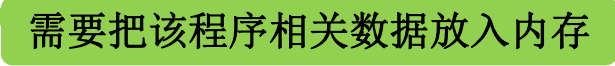

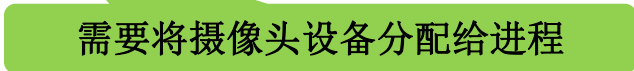
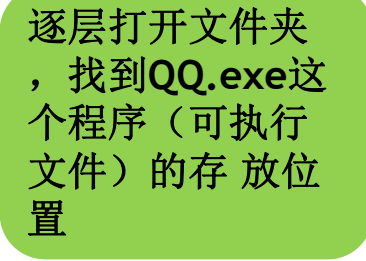
- **实现了对计算机资源的抽象**

裸机上覆盖上一层I/O设备管理软件  
将I/O设备抽象为一组数据结构以及一  
组I/O操作命令



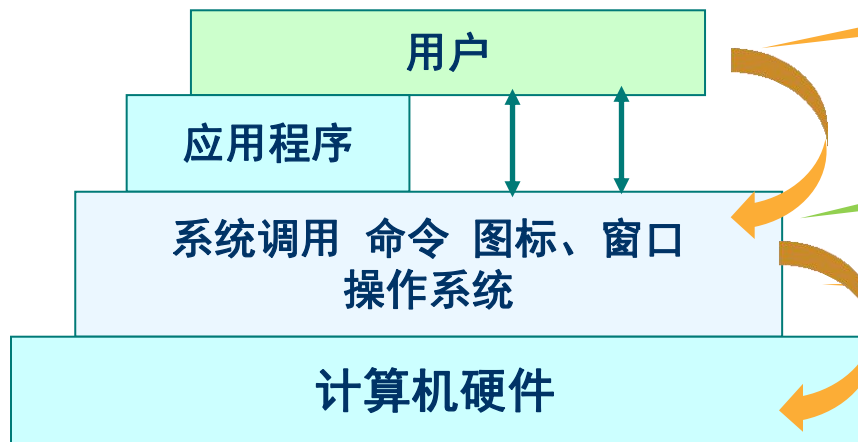


# 作为系统资源的管理者

- 执行一个程序前需要将该程序**放到内存中**，才能被CPU处理
  - 用QQ和朋友视频聊天的过程：
    - Step 1: 在各个文件夹中找到 QQ 安装的位置 (D:/Tencent/QQ/Bin)
    - Step 2: 双击打开 QQ.exe 
    - Step 3: QQ 程序正常运行 
    - Step 4: 开始和朋友视频聊天 
- 

# 向上层提供方便易用的服务

**封装思想：**操作系统把一些硬件功能封装成简单易用的服务，使用户能更方便地使用计算机，用户无需关心底层硬件的原理，只需要对操作系统发出命令即可

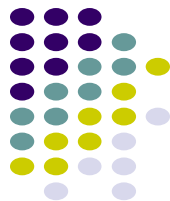


我需要执行A操作

在硬件之上安装了操作系统，操作系统对外提供友好的交互接口

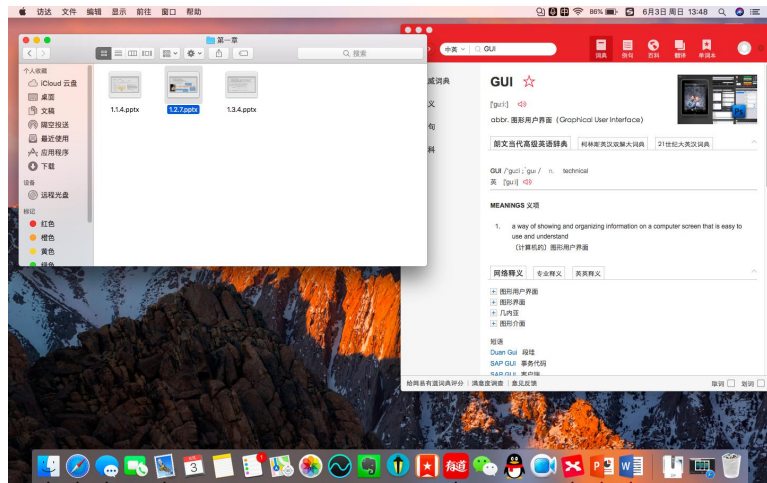
01010111101110111010101

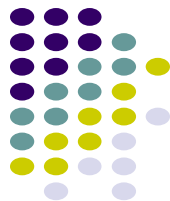
硬件只听得懂二进制指令，如：  
01010111101110111010101



# 向上层提供方便易用的服务

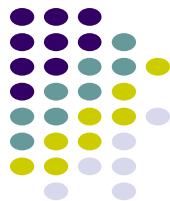
- GUI: 图形化用户接口, 用户可以使用形象的图形界面进行操作, 而不再需要记忆复杂的命令、参数。
- 例子: 在 Windows 操作系统中, 删除一个文件只需要把文件“拖拽”到回收站即可。





# 推动操作系统发展的主要动力

- 不断提高计算机资源利用率
- 方便用户
- 器件的不断更新换代
- 计算机体系结构的不断发展
- 不断提出新的应用需求

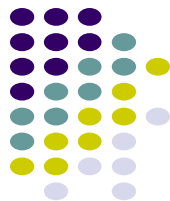


# 操作系统的发展过程

- 在20世纪50年代中期，出现了第一个简单的批处理OS；60年代中期开发出多道程序批处理系统；不久又推出分时系统，与此同时，用于工业和武器控制的实时OS也相继问世。20世纪70到90年代，是VLSI和计算机体系结构大发展的年代，导致了微型机、多处理机和计算机网络的诞生和发展，与此相应地，也相继开发出了微机OS、多处理机OS和网络OS，并得到极为迅猛的发展。



# 无OS



- 人工操作方式

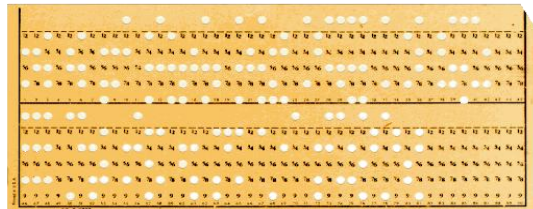
- 1946-50年代中：电子管时代，计算机速度慢，无操作系统，集中计算，计算机资源昂贵；

- 工作方式：

用户：既是程序员又是操作员；用户是专业人员；

输入输出：纸带或卡片；

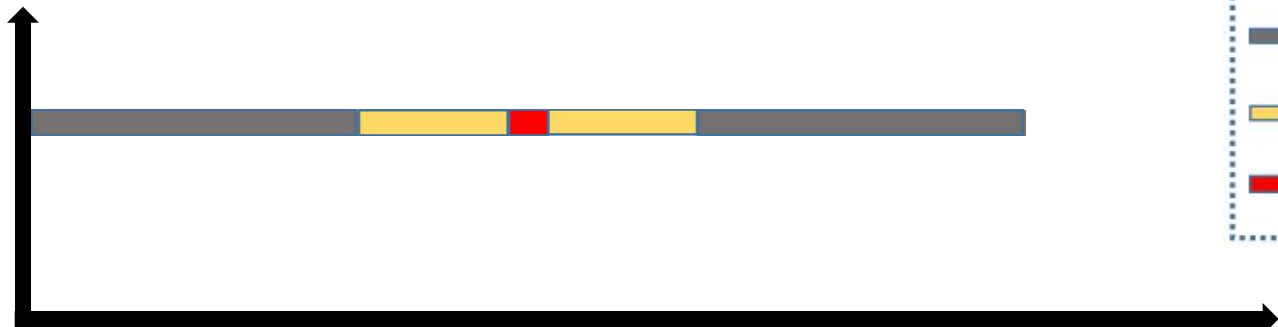
缺点：用户独占全机，  
CPU等待人工操作



# 无OS



作业

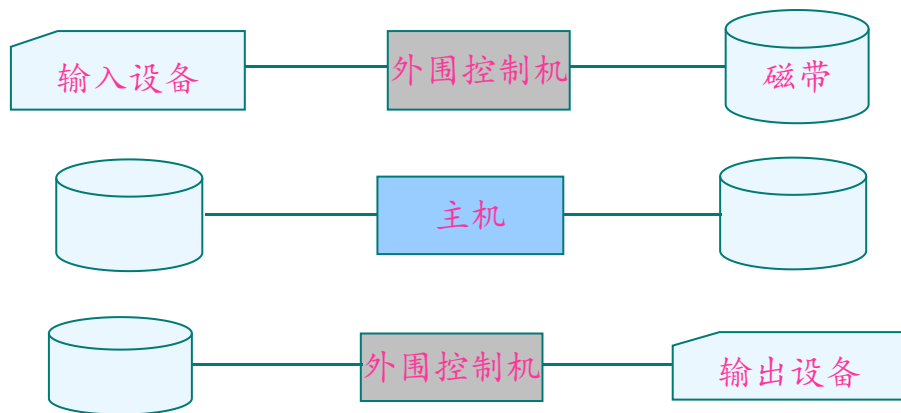


图示说明:

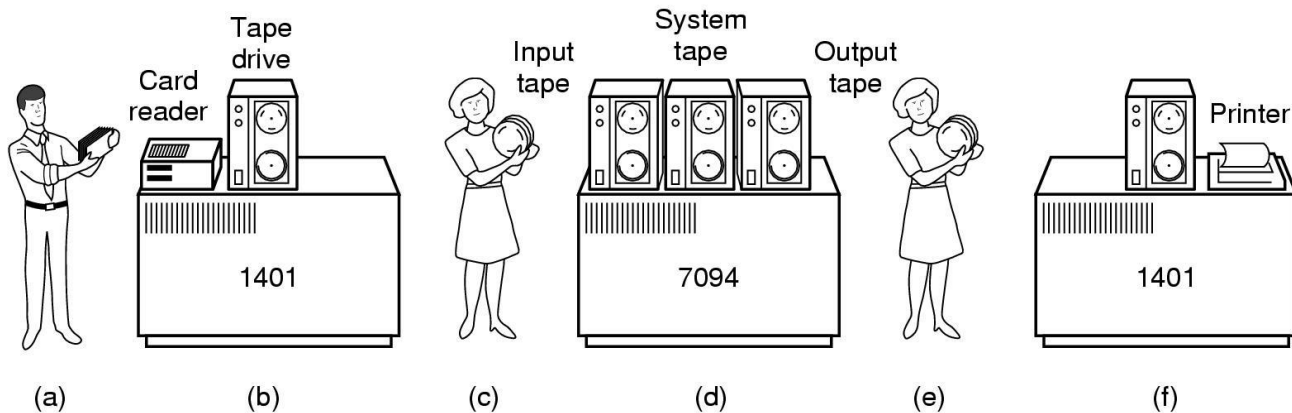
- 手工装/取纸带
- 从纸带机读/写
- 计算

时间

# 脱机I/O方式



减少了CPU的空闲时间;  
提高了I/O速度工作方式:



# 有OS



- 批处理系统 (Batch Processing System)
- 分时系统 (Time-Sharing System)
- 实时系统 (Real-Time System)



# 批处理系统

- 用户使用系统提供的作业控制语言（JCL）来描述自己对作业运行的控制意图，并将这些控制信息连同作业一起提交给计算机。
- 由OS去控制、调度各作业的运行并输出结果。
- 由于作业进入系统后用户不再干预，从而提高了效率。
- 提高系统资源的使用效率；提高作业吞吐量

设计目标：

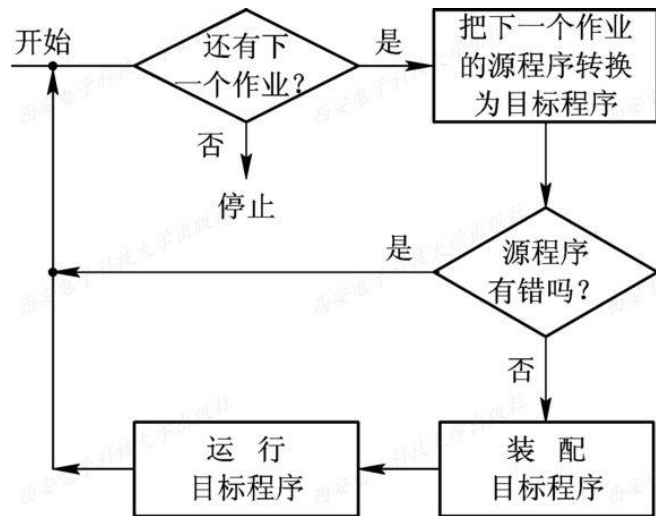
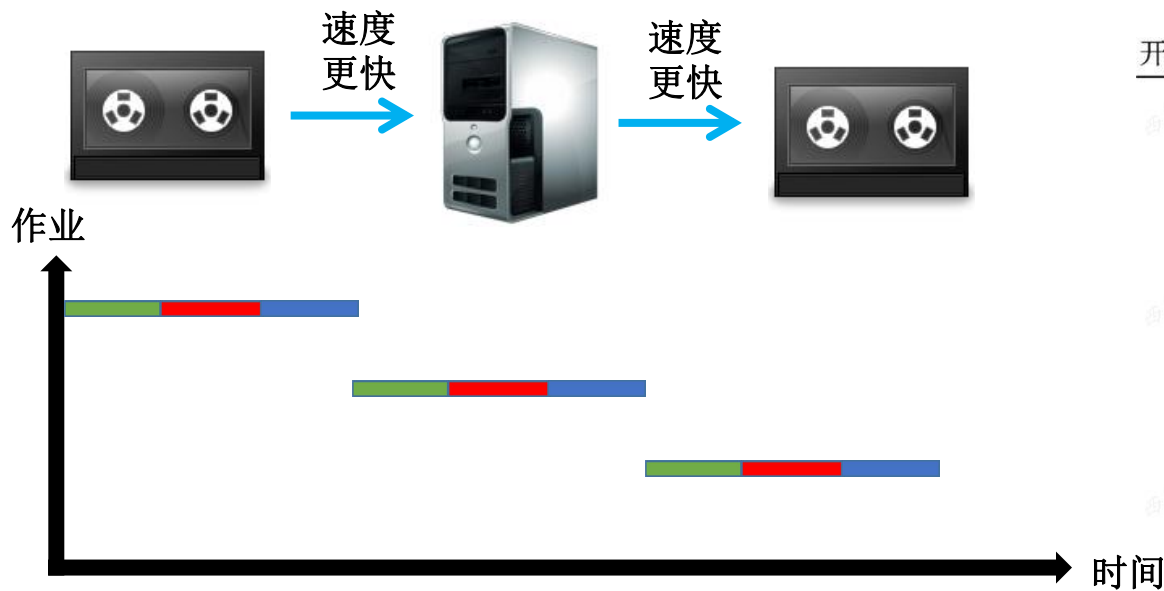
单道批处理系统；多道批处理系统

1、语文P34  
2、数学P25  
3、英语P36



# 单道批处理系统

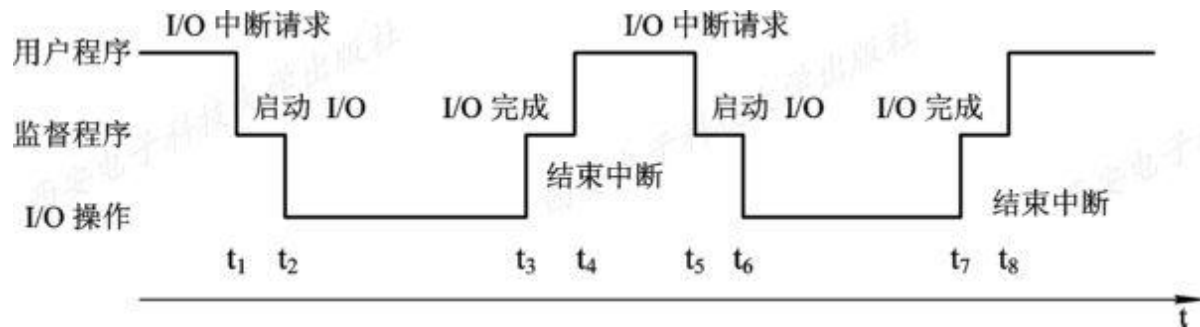
- 为实现对作业的连续处理，需要先把一批作业以脱机方式输入到磁带上，并在系统中配上监督程序(Monitor)，在它的控制下，使这批作业能一个接一个地连续处理。





# 单道批处理系统的缺点

- 系统中的**资源得不到充分的利用**。这是因为在内存中仅有一道程序，每逢该程序在运行中发出I/O请求后，CPU便处于**等待状态**，必须在其I/O完成后才继续运行。又因I/O设备的低速性，更使CPU的利用率显著降低。下图示出了单道程序的运行情况，从图可以看出：在 $t_2 \sim t_3$ 、 $t_6 \sim t_7$ 时间间隔内CPU空闲。





# 多道批处理系统

- 多道程序设计的基本概念
  - 在多道批处理系统中，用户所提交的作业都先存放在外存上并排成一个队列，称为“**后备队列**”；然后，由作业调度程序**按一定的算法**从后备队列中选择若干个作业调入**内存**，使它们共享CPU和系统中的各种资源
- 在20世纪60年代中期引入了多道程序设计技术，由此形成了**多道批处理系统**



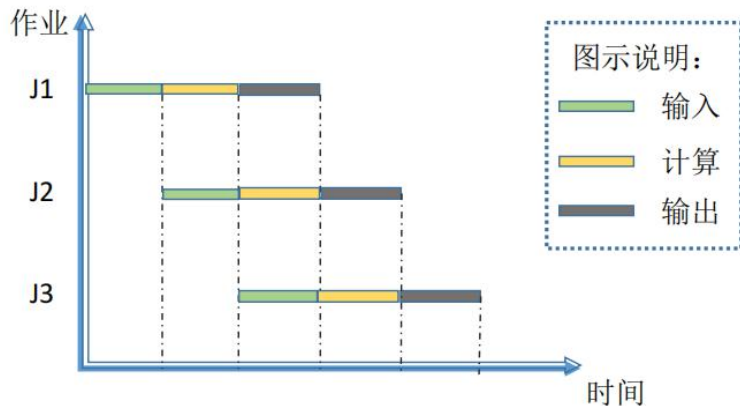
# 多道批处理系统



操作系统正式诞生，用于支持多道程序并发运行



每次往内存中读入多道程序

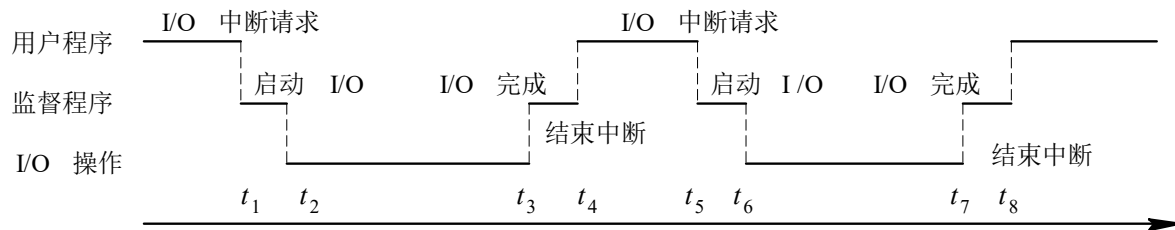


主要优点：多道程序**并发**执行，**共享**计算机资源。**资源利用率**大幅提升，CPU和其他资源更能保持“忙碌”状态，**系统吞吐量增大**。

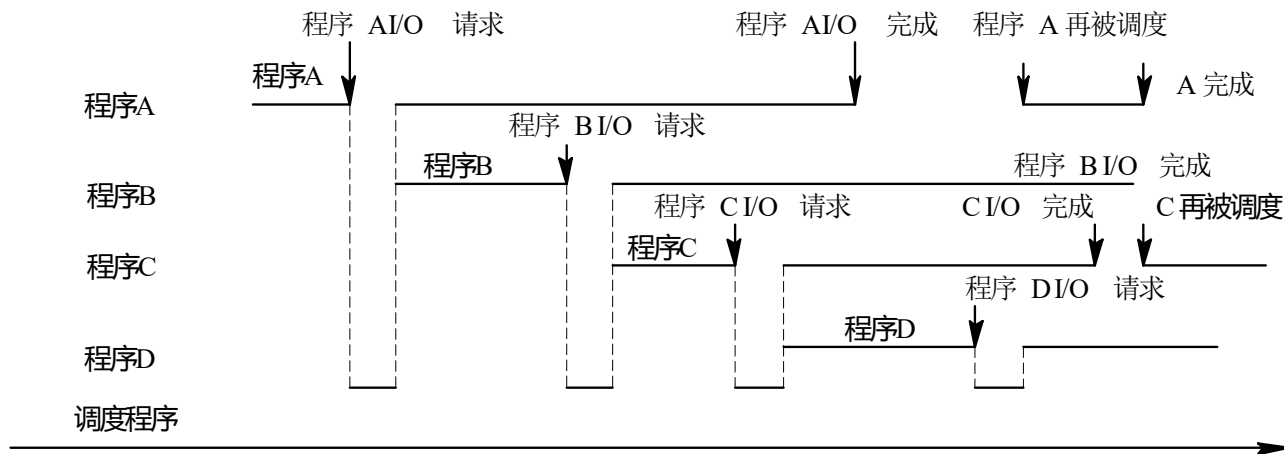
主要缺点：**平均周转时间长**，**没有人机交互功能**（用户提交自己的作业之后就只能等待计算机处理完成，中间不能控制自己的作业执行。eg：无法调试程序/无法在程序运行过程中输入一些参数）



# 单道程序、多道程序工作示例



(a) 单道程序运行情况



(b) 四道程序运行情况



# 多道批处理系统需要解决的问题

- **处理机争用问题**。既要能满足各道程序运行的需要，又要能提高处理机的利用率
- **内存分配和保护问题**。系统应能为每道程序分配必要的内存空间，使它们“各得其所”，且不会因某道程序出现异常情况而破坏其它程序
- **I/O设备分配问题**。系统应采取适当的策略来分配系统中的I/O设备，以达到既能方便用户对设备的使用，又能提高设备利用率的目的



# 多道批处理系统需要解决的问题

- **文件的组织和管理问题**。系统应能有效地组织存放在系统中的大量的程序和数据，使它们既便于用户使用，又能保证数据的安全性
- **作业管理问题**。系统中存在着各种作业(应用程序)，系统应能对系统中所有的作业进行合理的组织，以满足这些作业用户的不同要求
- **作业管理问题**。系统中存在着各种作业(应用程序)，系统应能对系统中所有的作业进行合理的组织，以满足这些作业用户的不同要求



# 分时系统引入

- 如果说推动多道批处理系统形成和发展的主要动力是提高资源利用率和系统吞吐量，那么，推动分时系统形成和发展的主要动力，则是为了满足用户对人一机交互的需求，由此形成了一种新型OS。用户的需求具体表现在以下几个方面
  - 人机交互(边运行边调试)
  - 共享主机(设备昂贵)

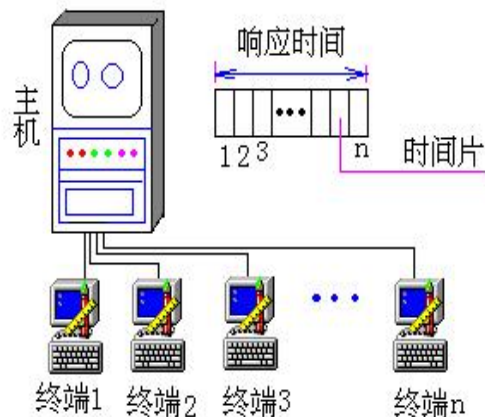


# 分时系统

- 一台计算机连接多个终端，用户通过各自的终端把作业送入计算机；计算机又通过终端向各个用户报告其作业的运行情况
  - 计算机以**时间片**为单位**轮流为各个用户/作业服务**
- 终端用户服务，并能及时地对用户服务请求予以响应

**目标：**对用户的请求及时响应；

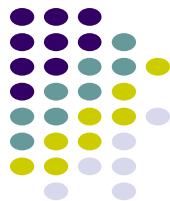
尽量提高系统资源的利用率





# 分时系统实现中的关键问题

- 在多道批处理系统中，用户无法与自己的作业进行交互的主要原因是：作业都先驻留在外存上，即使以后被调入内存，也要经过较长时间的等待后方能运行，用户无法与自己的作业进行交互
  - 及时接收——多路卡（多个I/O端口）
  - 及时处理——作业直接进内存，时间片轮转（分时技术）



# 分时系统的特征

- 分时系统与多道批处理系统相比，具有非常明显的不同特性，可以归纳成以下四个方面
  - 多路性：即同时性，宏观上同时，微观上轮流
  - 独立性：每个用户感觉独占主机
  - 及时性：较短时间响应(1-3秒)
  - 交互性：用户请求多方面服务





# 实时系统

- 提高系统的响应时间，对随机发生的外部事件作出**及时**响应并在**规定的时间内**对其进行处理

- 实时控制系统

要求计算机能尽快处理测量系统测得的数据，以尽快实施响应控制。如：工业控制；导弹发射；飞机飞行

- 实时信息系统

要求计算机能对终端设备发来的服务请求及时予以正确的回答。如：订票系统；股票交易系统



# 实时任务

- 按任务执行时是否呈现周期性来划分
  - 周期性实时任务
  - 非周期性实时任务
- 都必须联系着一个截止时间(Deadline)
  - 开始截止时间——任务在某时间以前必须开始执行
  - 完成截止时间——任务在某时间以前必须完成



# 实时任务

- 根据对截止时间的要求来划分

如：导弹控制系统  
， 自动驾驶系统

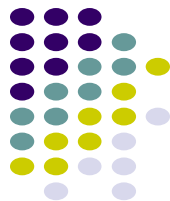
- **硬实时任务**(hard real-time task)。系统必须满足任务对截止时间的要求，否则可能出现难以预测的结果
- **软实时任务**(Soft real-time task)。它也联系着一个截止时间，但并不严格，若偶尔错过了任务的截止时间，对系统产生的影响也不会太大

如：12306订票系统

# 实时系统的特征

- 快速的响应时间
- 有限的交互能力
- 高可靠性





# 三种基本操作系统的比较

	多路性	独立性	及时性	交互性	可靠性
批处理系统	无	无	差 (天, 时)	差	一般
分时系统	多终端 服务	有	好 (分, 秒)	好	可靠
实时系统	多路采集、多路控制	有	最好 (ms, $\mu$ s)	一般	高度可靠



## 其他几种操作系统

- 网络操作系统：是伴随着计算机网络的发展而诞生的，能把网络中各个计算机有机地结合起来，实现数据传送等功能，**实现网络中各种资源的共享（如文件共享）和各台计算机之间的通信**。（如：Windows NT 就是一种典型的网络操作系统，网站服务器就可以使用）
- 分布式操作系统：主要特点是**分布性和并行性**。系统中的各台计算机地位相同，**任何工作都可以分布在这些计算机上，由它们并行、协同完成这个任务**。
- 个人计算机操作系统：如 **Windows XP、MacOS**，方便个人使用

## OS的发展与分类

手工操作阶段

缺：人机速度矛盾

批处理阶段

单道批处理系统（引入脱机输入输出技术）

优：缓解人机速度矛盾

缺：资源利用率依然很低

多道批处理系统（操作系统开始出现）

优：多道程序并发执行，资源利用率高

缺：不提供人机交互功能

分时操作系统

优：提供人机交互功能

缺：不能优先处理紧急任务

实时操作系统

~~硬实时系统~~ **任务** 必须在绝对严格的规定时间内完成处理

~~软实时系统~~ **任务** 能接受偶尔违反时间规定

优：能优先处理紧急任务

网络操作系统

分布式操作系统

个人计算机操作系统



# 微机操作系统的发展

- 单用户单任务操作系统
  - CP/M
  - MS-DOS
- 单用户多任务操作系统
  - 只允许一个用户上机，但允许用户把程序分为若干个任务，使它们并发执行，从而有效地改善了系统的性能
  - Windows
- 多用户多任务操作系统
  - 多用户通过各自终端使用同一台机器，UNIX OS，LINUX OS



# CP/M操作系统界面



CP/M系统由Digital Research公司（1991年被Novell兼并）在1974年开发。在70年代，它成为被广泛应用的操作系统。通过它的命令行选项，人们开始了解操作系统。微软20年后推出的DOS系统便是在它的基础上开发的。

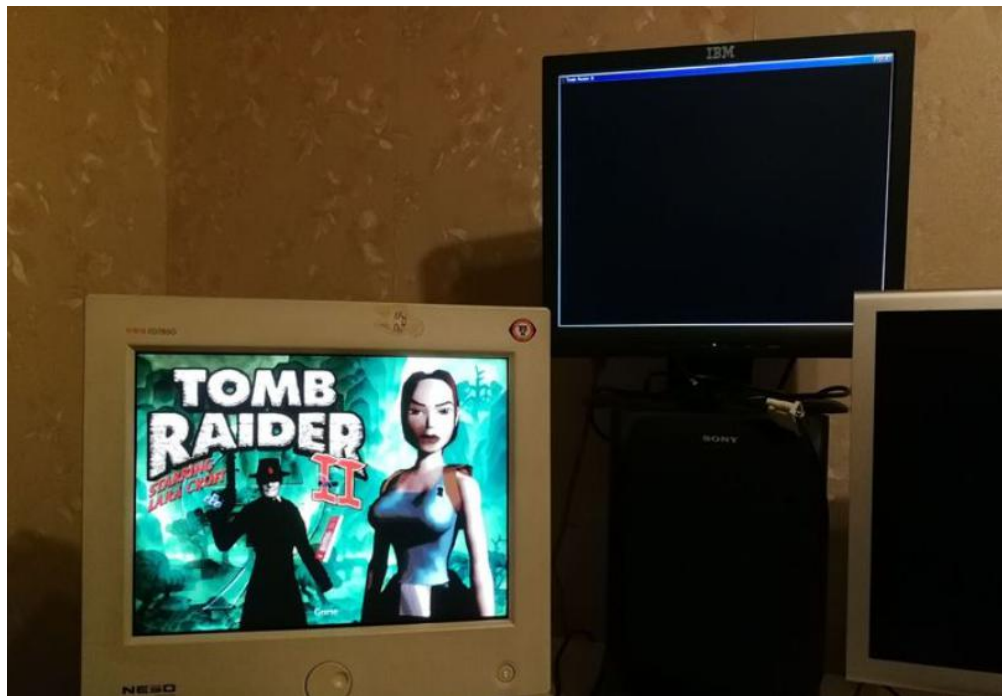


# MS-DOS界面

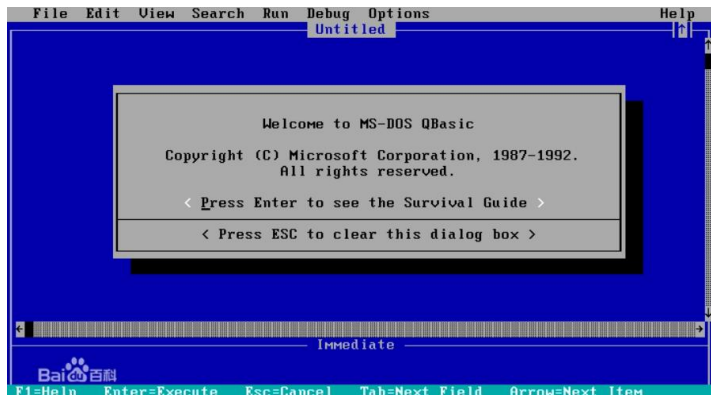
DOS又称磁盘操作系统。DOS命令行模式足足统治了系统市场15年（1981 到 1995 ）。若是把部分以DOS 为基础的 微软Windows 版本，如 Windows 95、98 和 Me 等都算进去的话，那么其商业寿命可以算到 2000 年。



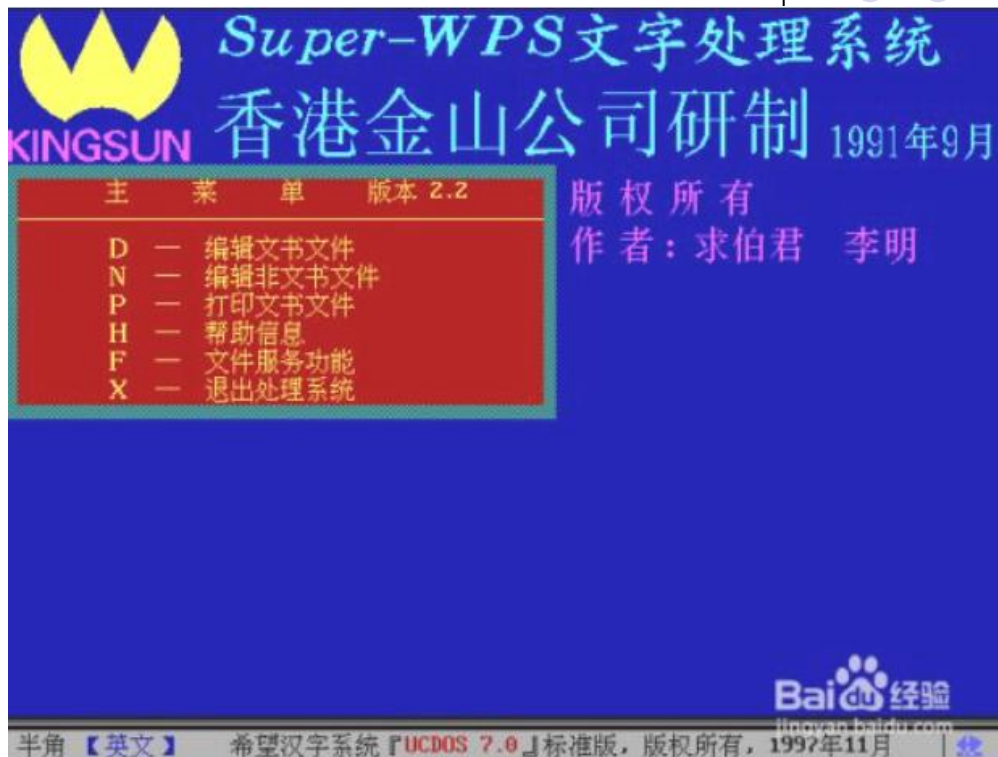
# DOS能做什么



# DOS能做什么



basic编程



wps



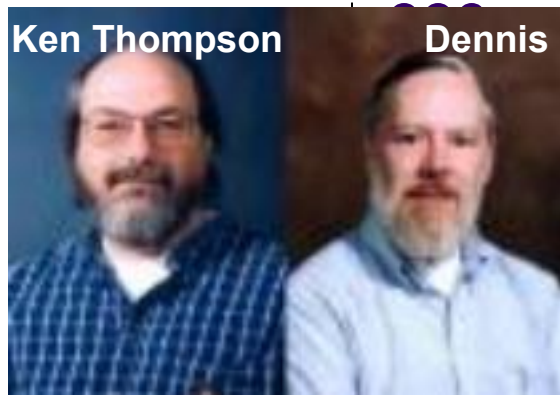
# Windows 95/98

Windows 95是微软1995年推出的操作系统，它第一次抛弃了对16位x86的支持。同时，Windows 95首次加入了开始菜单和任务栏两项功能。1998年微软推出了改进版 Windows 98



# UNIX的诞生

- 1965年Multics项目启动。MIT、贝尔实验室、通用电气公司参与。
- 1969年贝尔实验室退出。Dennis与Ken Thompson在PDP-7机上运行为Multics设计的“空间旅行”游戏。开发了浮点运算软件包、显示驱动、文件系统、实用程序、shell、汇编程序
- 1970年Unix诞生。
- Dennis用C重写。



# Dennis M. Ritchie---Unix之父



- Dennis于1967年加入贝尔实验室。
- 加入贝尔实验室不久，Dennis参与了Multics项目  
Multics项目为后来UNIX的产生打下了许多技术基础。
- Dennis除了与Ken Thompson发明与实现了UNIX操作系统之外，还是著名C语言的发明人。C语言来源于Thompson实现的B语言。C语言发明后，UNIX被用C来重写，从而使得UNIX的可移植性极大的提高。
- 1983年Dennis Ritchie和Ken Thompson一同被授予图灵奖

# Linux操作系统

- Linux的原型——Minix
- Minix的名称取自英语Mini UNIX, (约300MB)
- 全部的程序码共约12,000行
- 全套Minix除了启动的部分以汇编语言编写以外，其他大部份都是纯粹用C语言编写。分为：内核、内存管理及档案管理三部分



Andrew S. Tanenbaum



# Linux的诞生



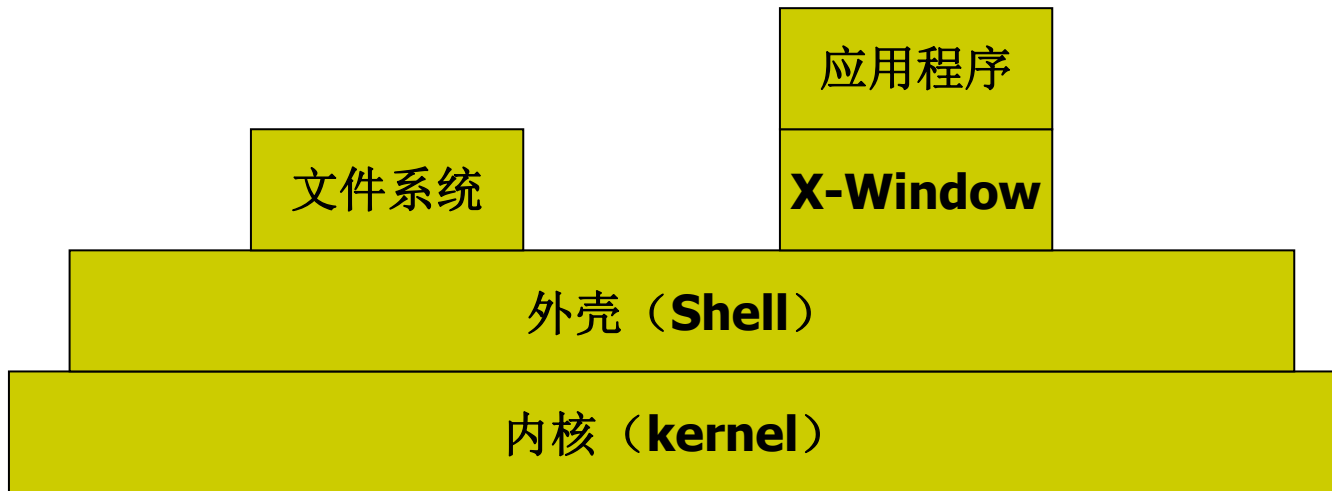
- Minix最有名的学生用户是Linus Torvalds，他在芬兰的赫尔辛基大学用Minix操作平台建立了一个新的操作系统的内核，他把它叫做Linux
- 1990年秋天， Linus开发了第一个程序，包括两个进程，向屏幕上写字母A和B，定时器切换。此外，从modem上接发信息的程序以及显示器、键盘、modem的驱动程序，文件系统
- 有了进程切换、文件系统、驱动程序，OS原型出现了。



# Linux内核的发展

- 1991年10月,芬兰赫尔辛基大学的学生Linus Torvalds为改进MINIX操作系统开发了一种类似Unix的操作系统,叫linux,最初发布的版本是0.02版
- 1994年,发布正式的1.0版本,linux开始成为一个比较完善的操作系统,并逐渐为世人所知
- 一些软件公司相继开发出自己的linux系统,如RedHat linux、RedFlag linux等
- 大量的软件专家和linux爱好者不断地提高和改进linux内核功能
- 应用软件厂商开发出大量基于Linux的应用软件

# Linux操作系统的构成





## 1.3 操作系统的基本特性

- **并发**

- 并发：指两个或多个事件在同一时间间隔内发生
- 并行：指两个或多个事件在同一时刻发生

- **引入进程**

- 若对内存中的多个程序都分别建立一个进程，它们就可以并发执行，这样便能极大地提高系统资源的利用率，增加系统的吞吐量

**进程是在系统中能独立运行并作为资源分配的基本单位，由一组机器指令、数据和堆栈组成，是一个能独立运行的活动实体**



# 操作系统的基本特性——并发

- 并发 VS 并行

- eg: 假设小明和小红约同学出去玩，每人都有2个同学。任务1：和一号出去；任务2：和二号出去...



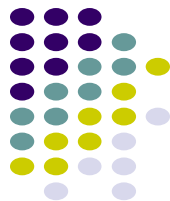
和一号、二号一起  
出去

**并行**：同一时刻同时进行两个任务



8点~9点：一号  
9点~10点：二号  
10点~11点：一号  
.....

**并发**：宏观上看，这一天小红在同时进行两个任务。微观上看，在某一时刻，小红最多正在进行一个任务



# 操作系统的基本特性——并发

- 并发：指两个或多个事件在同一时间间隔内发生。这些事件**宏观上是同时发生的**，但**微观上是交替发生的**
- **操作系统的并发性**指计算机系统中“**同时**”运行着**多个程序**，这些程序宏观上看是同时运行着的，而微观上看是交替运行的。
- 操作系统就是伴随着“多道程序技术”而出现的。因此，**操作系统和程序并发是一起诞生的**。
- 注意（重要考点）：
  - **单核CPU**同一时刻只能执行一个程序，各个程序只能并发地执行
  - **多核CPU**同一时刻可以同时执行多个程序，多个程序可以并行地执行，比如Intel的第八代 i3 处理器就是 4 核CPU，意味着可以并行地执行4个程序

即使是对于**4核CPU**来说，只要有**4个以上**的程序需要“**同时**”运行，那么并发性依然是必不可少的，因此**并发性是操作系统一个最基本的特性**





# 操作系统的基本特性——共享

## ● 共享

- 指系统中的资源供内存中的多道程序所共同使用

系统中的某些资源，虽然可以提供给多个进程使用，但一个时间段内只允许一个进程访问该资源

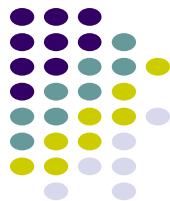
分类 { 互斥共享方式： 例 打印机  
同时访问方式： 例 磁盘

- 临界资源：一段时间内只允许一个进程访问的资源，如打印机

系统中的某些资源，允许一个时间段内由多个进程“同时”对它们进行访问

互斥共享方式：使用QQ和微信视频。同一时间段内摄像头只能分配给其中一个进程。

同时访问方式：使用QQ发送文件A，同时使用微信发送文件B。宏观上看，两边都在同时读取并发送文件，说明两个进程都在访问硬盘资源，从中读取数据。微观上看，两个进程是交替着访问硬盘的



# 并发和共享的关系

- **并发性**指计算机系统中同时存在着多个运行着的程序。
- **共享性**是指系统中的资源可供内存中多个并发执行的进程共同使用
- 使用QQ发送文件A，同时使用微信发送文件B。
  - 1. 两个进程正在并发执行（**并发性**）
  - 2. 需要共享地访问硬盘资源（**共享性**）

如果失去并发性，则系统中只有一个程序正在运行，则共享性失去存在的意义

如果失去共享性，则QQ和微信不能同时访问硬盘资源，就无法实现同时发送文件，也就无法并发

并发性



共享性





# 操作系统的基本特性——虚拟

- 虚拟

指通过某种技术把一个物理实体变成若干个逻辑上的对应物

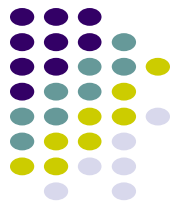
- 时分复用技术

- 虚拟处理机技术
- 虚拟设备技术

- 空分复用技术

- 计算机中把空分复用技术用于对存储空间的管理，用以提高存储空间的利用率，如磁盘分区

# 操作系统的基本特性——虚拟



- 背景知识：一个程序需要放入内存并给它分配CPU才能执行



GTA5需要4GB的运行内存，QQ 需要256MB的内存，迅雷需要256MB的内存，网易云音乐需要256MB的内存

.....

我的电脑：4GB内存

问题：这些程序同时运行需要的内存远大于4GB，那么为什么它们还可以在我的电脑上同时运行呢？答：这是虚拟存储器技术。实际只有4GB的内存，在用户看来似乎远远大于4GB



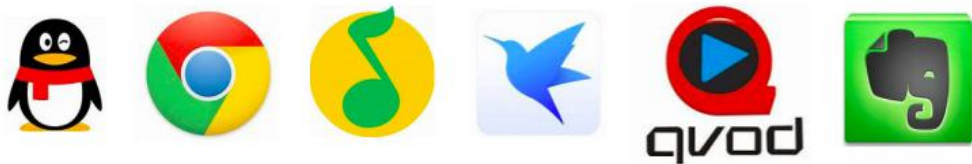
六六六六六六六六

虚拟技术中的“空分复用技术”



# 操作系统的基本特性——虚拟

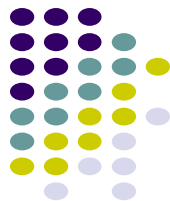
- 背景知识：一个程序需要放入内存并给它分配CPU才能执行
- 某单核CPU的计算机中，用户打开了以下软件。。。



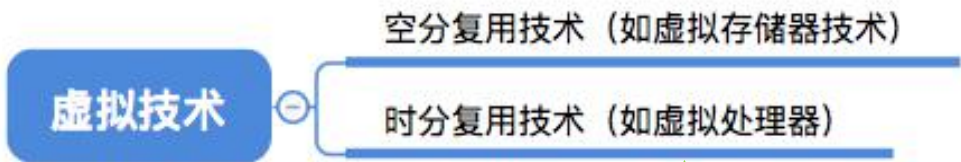
问题：既然一个程序需要被分配CPU才能正常执行，那么为什么单核CPU的电脑中能同时运行这么多个程序呢？

答：这是虚拟处理器技术。实际上只有一个单核CPU，在用户看来似乎有 6个CPU在同时为自己服务

虚拟技术中的“时分复用技术”



# 操作系统的基本特性——虚拟



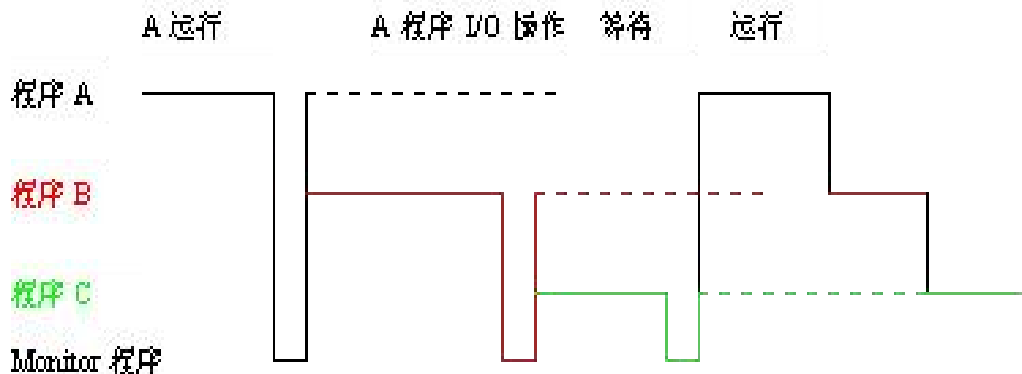
显然，如果失去了并发性，则一个时间段内系统中只需运行一道程序，那么就失去了实现虚拟性的意义了。因此，**没有并发性，就谈不上虚拟性**



# 操作系统的基本特性

- 异步

- 由于资源有限，进程的执行不是一贯到底的，系统中并发执行的多道程序“走走停停”，以不可预知的速度向前推进。

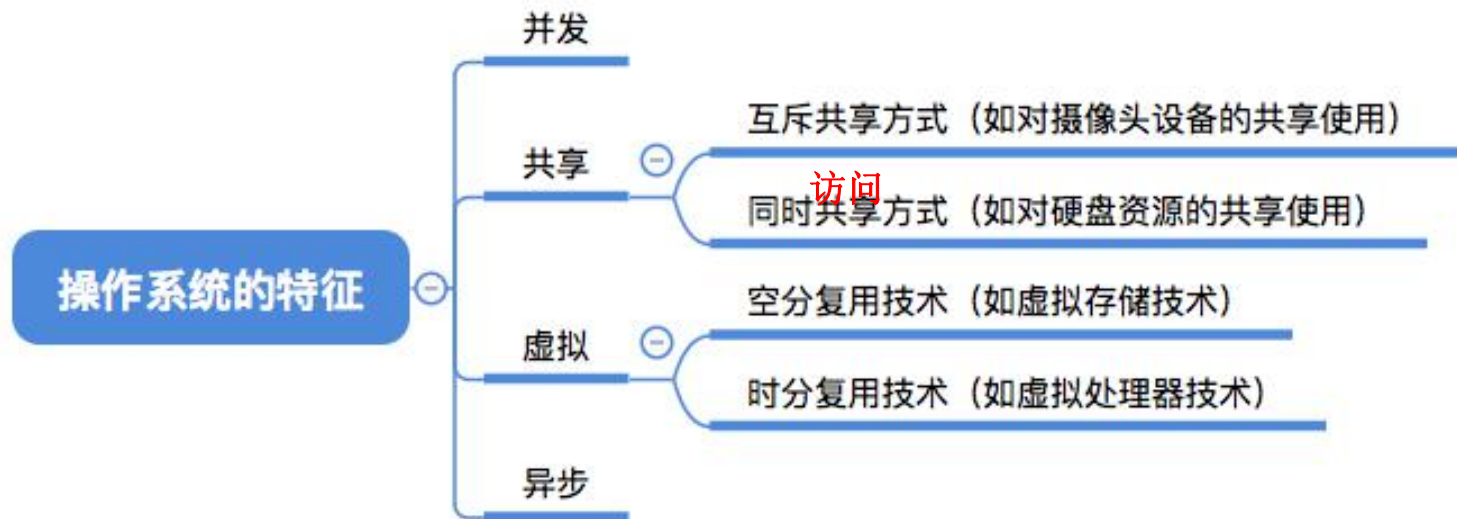




# 操作系统的基本特性——异步

- 由于资源有限，进程的执行不是一贯到底的，系统中并发执行的多道程序“走走停停”，以不可预知的速度向前推进。
- 由于并发运行的程序会**争抢着使用系统资源**，而系统中的资源有限，因此进程的执行不是一贯到底的，而是走走停停的，以不可预知的速度向前推进
- 如果失去了并发性，即系统只能串行地运行各个程序，那么每个程序的执行会一贯到底。**只有系统拥有并发性，才有可能导致异步性。**

# 知识回顾

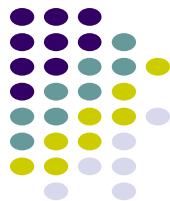


重要考点：

理解并发和并行的区别

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，因此并发和共享是操作系统的两个最基本的特征



## 1.4 操作系统的主要功能

- 处理机管理功能
- 存储器管理功能
- 设备管理功能
- 文件管理功能
- 用户接口

OS是直接控制和管理计算机硬件、软件资源，合理地各类作业进行调度，以方便用户使用的程序集合



# 处理机管理功能



- 进程控制

- 在传统的多道程序环境下，要使作业运行，必须先为它创建一个或几个进程，并为之分配必要的资源。当进程运行结束时，立即撤消该进程，以便能及时回收该进程所占用的各类资源。进程控制的主要功能是为作业创建进程、撤消已结束的进程，以及控制进程在运行过程中的状态转换

# 处理机管理功能

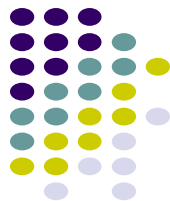


- 进程同步
  - 进程互斥方式：诸进程在对临界资源进行访问时，应采用互斥方式
  - 进程同步方式：在相互合作去完成共同任务的诸进程间，有同步机构对他们的执行次序加以协调

# 处理机管理功能



- 进程通信
  - 在多道程序环境下，为了加速应用程序的运行，应在系统中建立多个进程，并且再为一个进程建立若干个线程，由这些进程(线程)相互合作去完成一个共同的任务。而在这些进程(线程)之间，又往往需要交换信息
- 进程调度
  - 作业调度
  - 进程调度



# 存储器管理功能

- 内存分配
- 主要任务：
  - 为每道程序分配内存空间，使它们“各得其所”
  - 提高存储器的利用率，尽量减少不可用的内存空间(碎片)
  - 允许正在运行的程序申请附加的内存空间，以适应程序和数据动态增长的需要

静态分配方式、动态分配方式

# 存储器管理功能



- 内存保护

- 内存保护的主要任务，是确保每道用户程序都只在自己的内存空间内运行，彼此互不干扰
- 绝不允许用户程序访问操作系统的程序和数据，也不允许用户程序转移到非共享的其他用户程序中去

# 存储器管理功能



- 地址映射
  - 能够将地址空间中的逻辑地址转换为内存空间中与之对应的物理地址
- 内存扩充
  - 借助虚拟存储技术，在逻辑上扩充内存容量，改善系统性能，让更多的用户程序并发运行



# 设备器管理功能

- 缓冲管理
  - 缓和CPU和I/O设备速度不匹配的矛盾
  - 设置缓冲区和缓冲机制，改善系统性能
- 设备分配
  - 设备控制表、控制器等，将系统资源按照某种策略进行分配
- 设备处理
  - 设备驱动程序，实现CPU和设备控制器之间的通信，完成制定的I/O操作



# 文件管理功能

- 存储空间管理
  - 为文件分配外存空间、提高外存利用率、提高文件存取速度
- 目录管理
  - 文件名、文件属性、物理位置等管理
  - 文件共享、目录查询，存取
- 文件的读写管理和保护
  - 通过文件读写指针进行读写，并为下一次读写做准备
  - 非法用户存取文件；防止不正确使用文件





# 现代操作系统的新功能

- 系统安全
  - 认证技术
    - 身份认证
  - 密码技术
    - 数据、资源加密保护、保护系统安全
  - 访问控制技术
    - 用户存取权限设置；文件属性保护，如只读文件
  - 反病毒技术
    - 防止病毒入侵



# 现代操作系统的新功能

- 网络功能和服务
  - 网络通信
    - 传输控制、差错控制、流量控制
  - 资源管理
    - 管理网络共享资源，保障数据安全，如打印机、硬盘等共享资源
  - 应用互操作
    - 在不同网络的互联网中提供互操作功能，实现信息互通，用户可访问不同网络中的文件



# 现代操作系统的新功能

- 支持多媒体
  - 接纳控制功能
    - 管理内存中的任务数目
  - 实时调度
    - 考虑进程调度策略及进程调度的接纳度，如图像更新周期必须在40ms之内
  - 多媒体文件的存储
    - 保证硬盘数据快速传输到输出设备，需要改进数据存储方式和寻道方式



## 1.5 OS结构设计

- 操作系统的结构设计经历了以下几代：
  - 传统的操作系统结构
  - 无结构操作系统
  - 模块化OS结构
  - 分层式OS结构
- 现代操作系统结构
  - 微内核的OS结构



# 无结构操作系统

- OS是由众多的过程直接构成，各过程之间可相互调用，但OS内部不存在任何结构，所以这种OS是无结构的，又称为整体系统结构

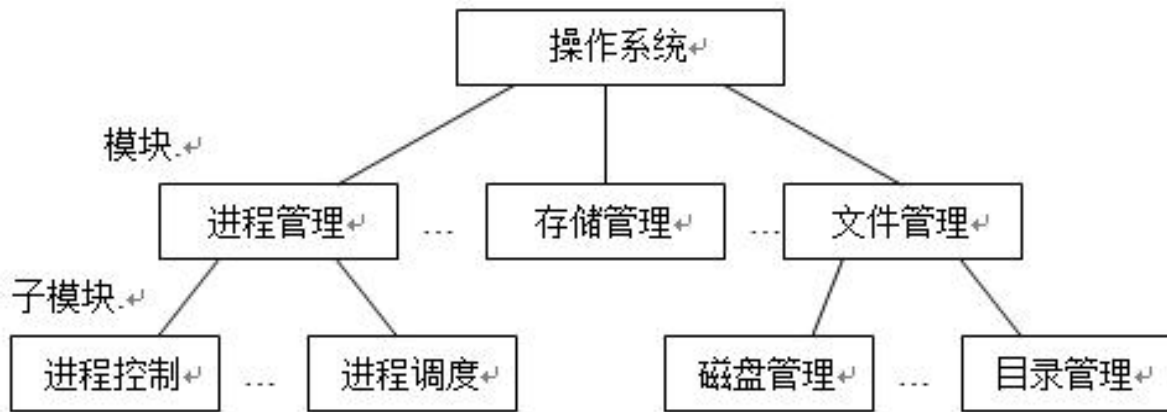
## 缺点

既庞大又杂乱，缺乏清晰的程序结构；程序错误多，调试难、阅读理解难、维护难。



# 模块化操作系统结构

- OS是采用“模块化程序设计”技术，按其功能划分为若干个独立的模块，管理相应的功能，同时规定好各模块之间的接口，以实现其交互，对较大模块又可按子功能进一步细分下去



# 模块化操作系统结构



## 优点

- ✓ 提高了OS设计的正确性、可理解性和可维护性
- ✓ 容易扩充，增强了OS的可适应性
- ✓ 加速了OS的开发过程

## 缺点

- ✓ 模块及接口划分较困难
- ✓ 从功能上划分模块，未区别共享资源和独占资源
- ✓ 由于管理的差异，使OS结构变得不够清晰



# 分层式操作系统结构

- 分层式OS结构是对模块化结构的一种改进，它按分层式结构设计的基本原则，将OS 划分为若干个层次，每一层都只能使用其底层所提供的功能和服务，从硬件开始，在其上面一层一层地自底向上增添相应功能的软件，这种OS结构称为分层式OS结构

## 特点：

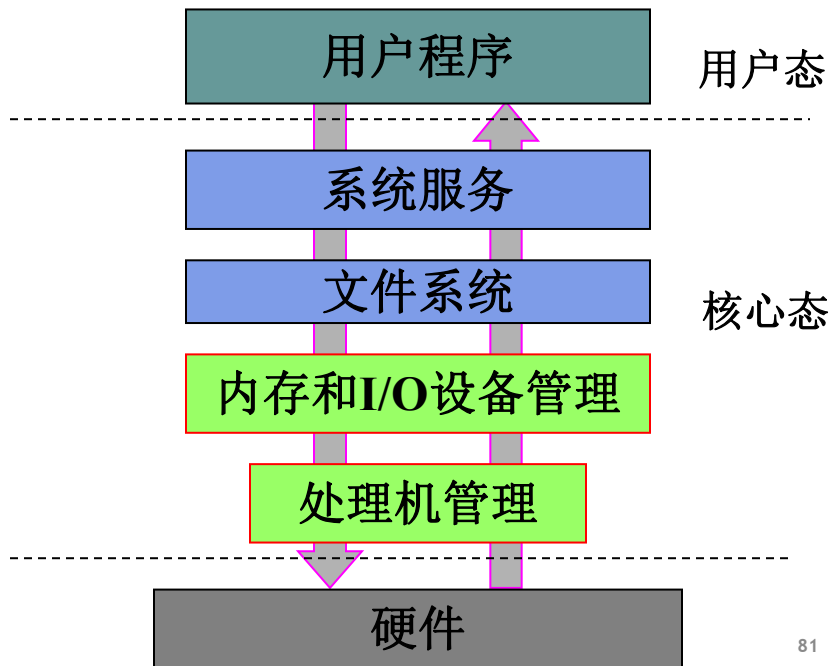
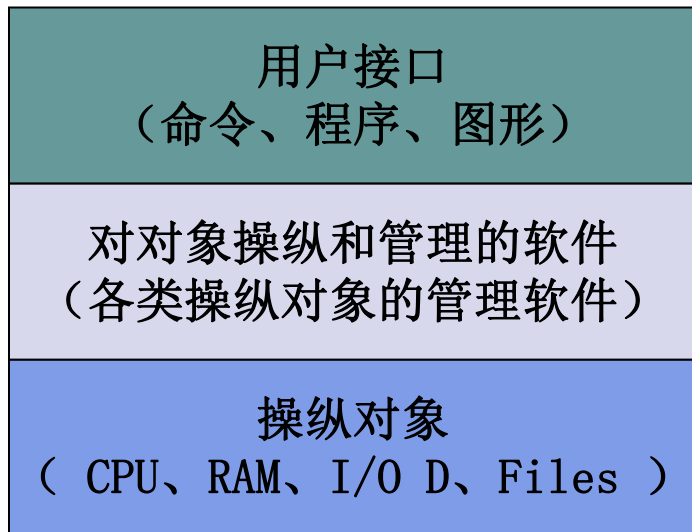
- ✓ 每一步设计都建立在可靠的基础上，结构更清晰
- ✓ 调试和验证更容易，正确性更高
- ✓ 缺点：系统效率降低了。





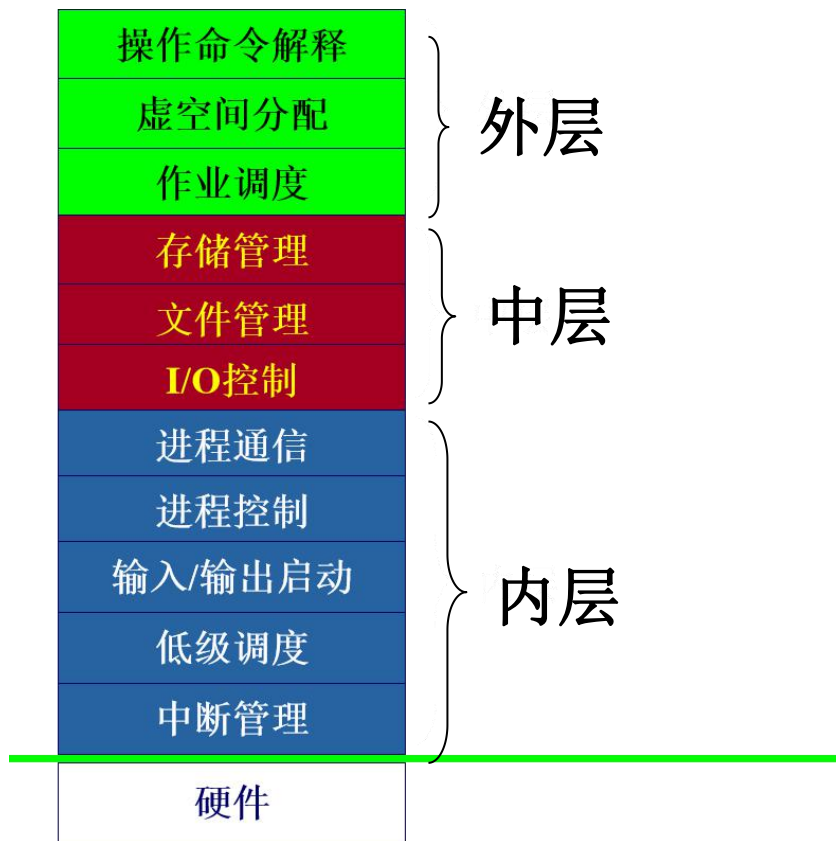
# 分层式操作系统结构

- **原理**——从资源管理观点出发，划分层次。各层模块间只能是单向调用关系，使模块间的调用变为有序性





# 层次的设置大致原则



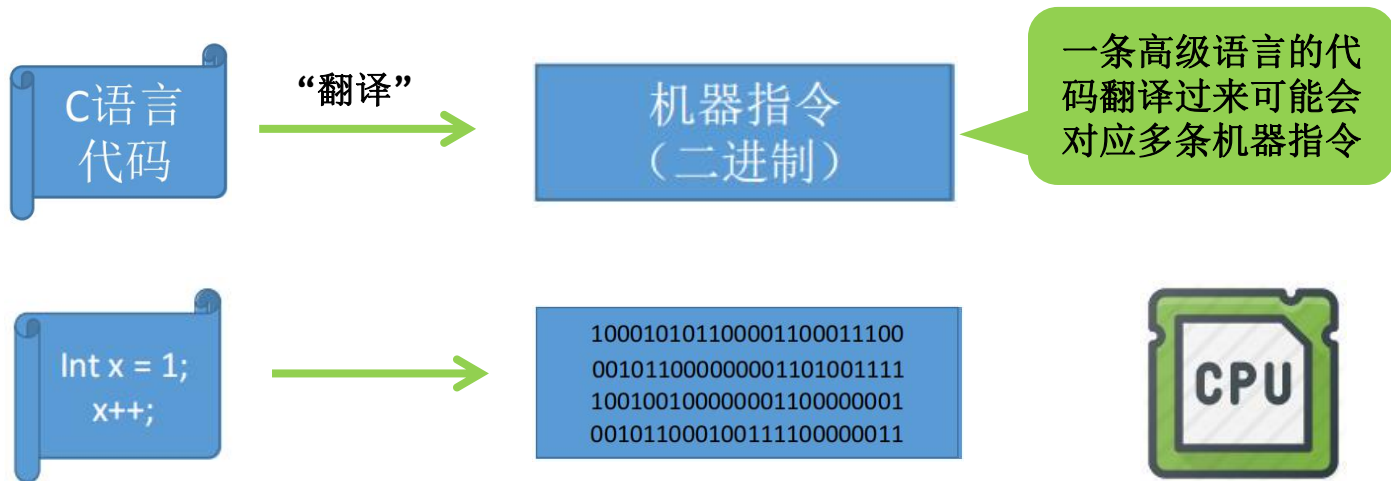
左图是荷兰科学家Dijkstra于1968年建造的第一个层次结构：SUE OS



# 分层结构的特点

- 优点：
  - 功能明确，调用关系清晰（高层对低层单向依赖），有利于保证设计和实现的正确性
  - 低层和高层可分别实现（便于扩充）；高层错误不会影响到低层；避免递归调用
- 缺点：各系统对具体划分多少层次有不同的看法。

# 程序是如何运行的？

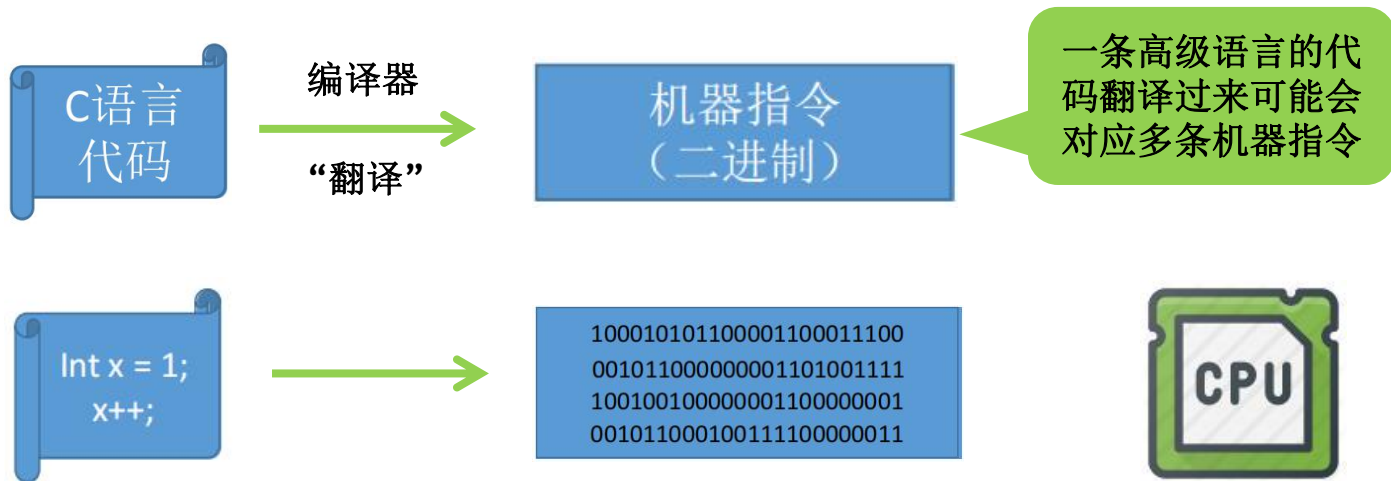
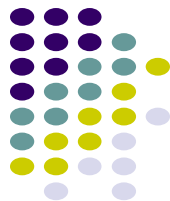


“指令”就是处理器（CPU）能识别、执行的最基本命令

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

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

# 程序是如何运行的？



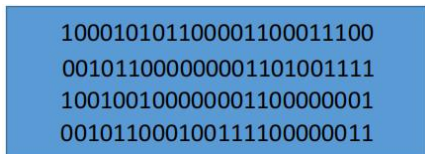
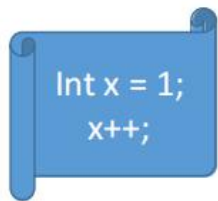
我们普通程序员写的程序就是“**应用程序**”

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



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

# 特权指令 v.s. 非特权指令



我们普通程序员写的程序就是“应用程序”

应用程序只能使用“非特权指令”，如：加法指令、减法指令等

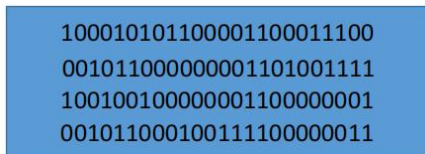
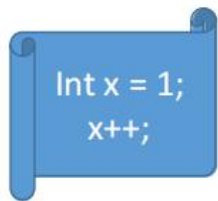
微软、苹果有一帮人负责实现操作系统，他们写的就是“内核程序”

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

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

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（核心态、管态） v.s. 用户态（目态）



CPU 能判断出指令类型，但是它怎么区分此时正在运行的是内核程序 or 应用程序？

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

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**

如I/O操作，内存清零

运算指令

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**”



# 内核态、用户态的切换

- **内核态→用户态**：执行一条**特权指令——修改PSW的标志位为“用户态”**，这个动作意味着操作系统将主动让出CPU使用权
- **用户态→内核态**：由**“中断”**引发，**硬件自动完成变态过程**，触发中断信号意味着操作系统将强行夺回CPU的使用权

除了非法使用特权指令之外，还有很多事件会触发中断信号。一个共性是，**但凡需要操作系统介入的地方，都会触发中断信号**





# 内核态、用户态 的切换

一个故事：

- ① 刚开机时，CPU 为“**内核态**”，操作系统内核程序先上CPU运行
- ② 开机完成后，用户可以启动某个应用程序
- ③ 操作系统内核程序在合适的时候主动让出 CPU，让该应用程序上CPU运行
- ④ 应用程序运行在“**用户态**”
- ⑤ 此时，一位猥琐黑客在应用程序中植入了一条特权指令，企图破坏系统...
- ⑥ CPU发现接下来要执行的这条指令是特权指令，但是自己又处于“用户态”
- ⑦ 这个非法事件会引发一个**中断信号**
- ⑧ “中断”使操作系统再次夺回CPU的控制权
- ⑨ 操作系统会对引发中断的事件进行处理，处理完了再把CPU使用权交给别的应用程序

操作系统内核在让出CPU之前，会用一条**特权指令**把**PSW**的标志位设置为“**用户态**”

**CPU检测到中断信号后**，会立即变为“**核心态**”，并停止运行当前的应用程序，转而运行处理中断信号的内核程序

# 操作系统的运行机制

简单了解程序的运行原理

高级语言编写代码——>机器指令

程序运行的过程就是CPU 执行指令的过程

两类程序

内核程序

应用程序

两类指令

特权指令

非特权指令

两种处理器状态

内核态/核心态

用户态/目态

内核

内核(Kernel)是操作系统最重要最核心的部分

由很多内核程序组成操作系统内核

如何变态?

内核态—>用户态

一条修改PSW的特权指令

用户态—>内核态

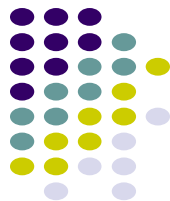
由中断引起, 硬件自动完成

only

only

only

# 操作系统体系结构



操作系统的体系结构

大内核/单内核/宏内核

微内核



# 操作系统的内核



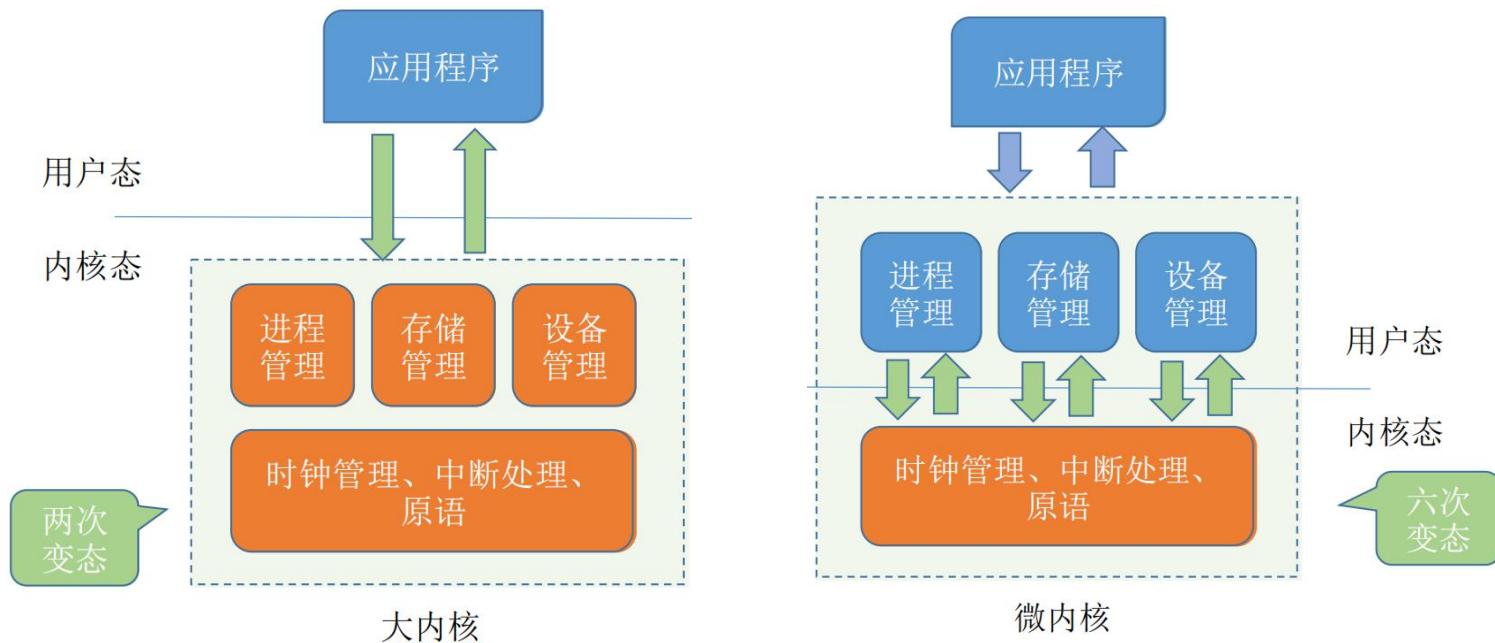


# 操作系统的内核

- **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**。



# 大内核 v.s. 微内核



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**



# 微内核的OS结构

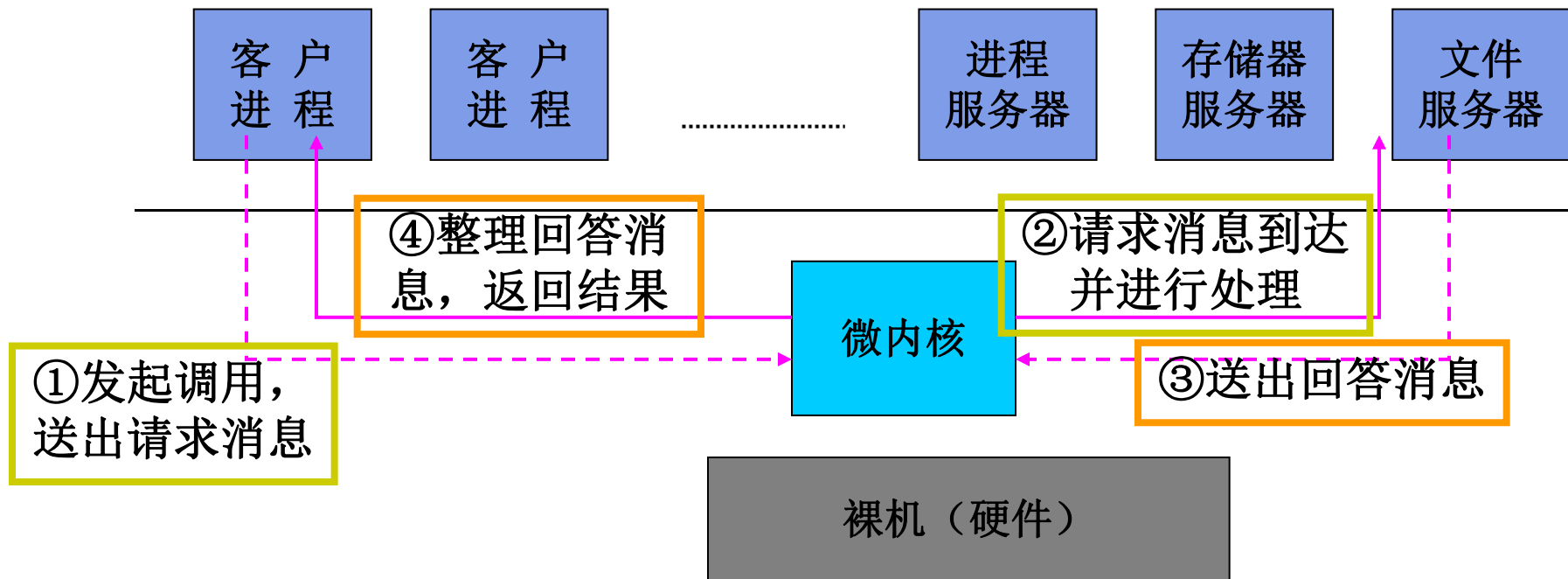
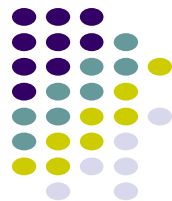
- 在OS内核中只留下一些最基本的功能，而将其他服务分离出去，由工作在用户态下的进程来实现，形成所谓“客户/服务器”模式。客户进程可通过内核向服务器进程发送请求，以获取OS的服务

## 特点

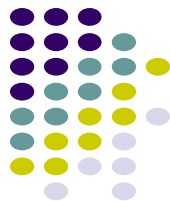
- ✓ 小而精练
- ✓ 系统的灵活性和可扩充性好
- ✓ 系统的可靠性高
- ✓ 适用于分布式系统

例，windows 2000/XP、UNIX、嵌入式OS

# 客户/服务器模式







# 客户/服务器模式

- 优点

- 充分模块化、减少系统的内存需求、高可移植性

- 缺点

- 各模块与微内核间通过通信机制交互，系统运行效率较低

**Windows采用改进的微内核机制**



# 微内核的基本功能

- 进程（线程）管理
  - 进程调度、进程切换、进程通信，处理及同步
  - 机制与策略分离
- 低级存储器管理
  - 低级存储器管理机制、用户空间的逻辑地址转为内存空间的物理地址
- 中断和陷入处理
  - 中断识别、保护、信息传递

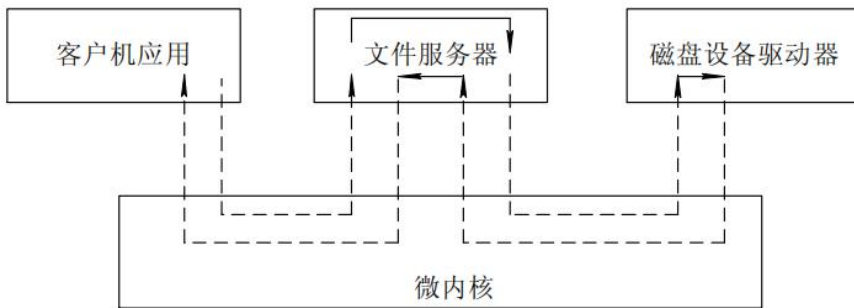
# 微内核OS的优/缺点

- 提高了系统的可扩展性
- 增强了系统的可靠性
- 可移植性强
- 提供了对分布式系统的支持
- 融入了面向对象技术
- 运行效率低



返回

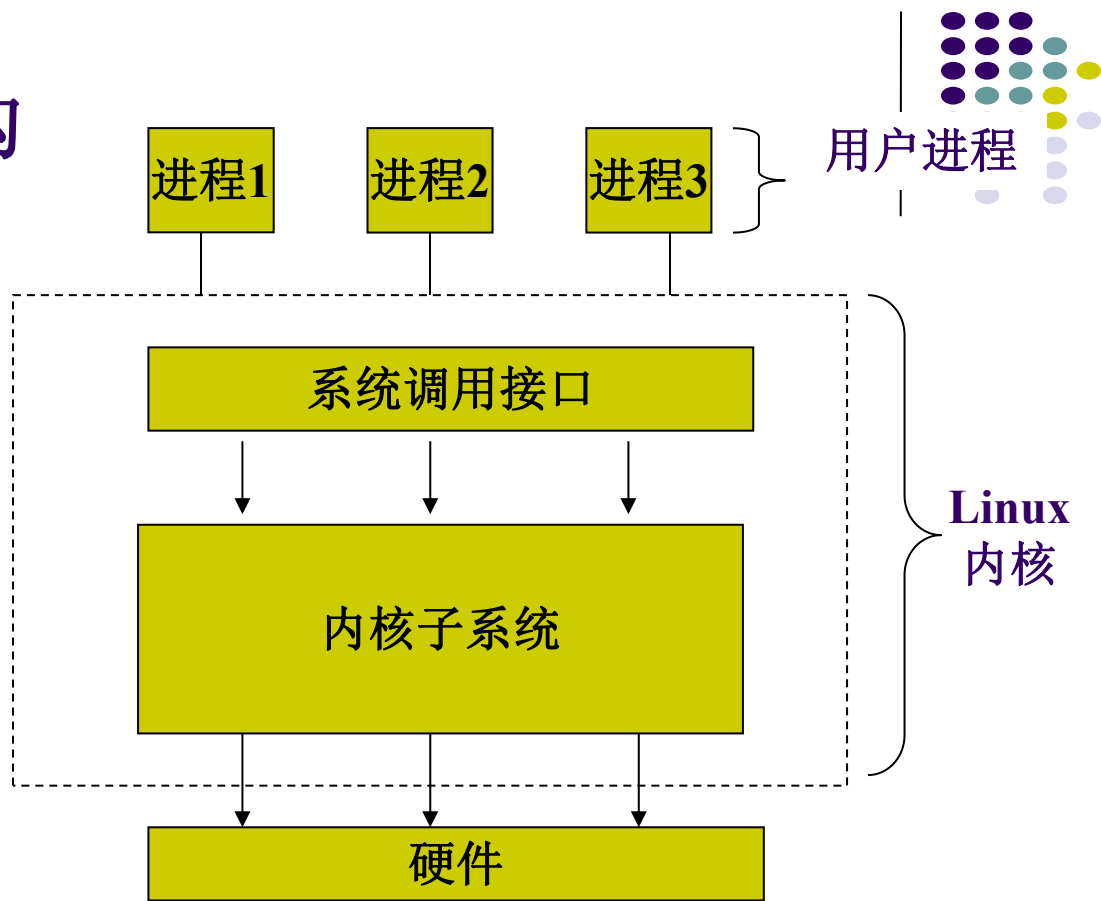
(a) 在整体式内核文件操作中的上下文切换



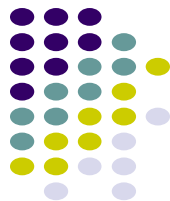
(b) 在微内核中等价操作的上下文切换

# 补充-linux内核结构

- Linux采用单内核机制



Linux内核在整个系统中的位置



# 操作系统内核



# 操作系统体系结构



## 操作系统的体系结构

大内核

将操作系统的主要功能模块都作为系统内核，运行在核心态

优点：高性能

缺点：内核代码庞大，结构混乱，难以维护

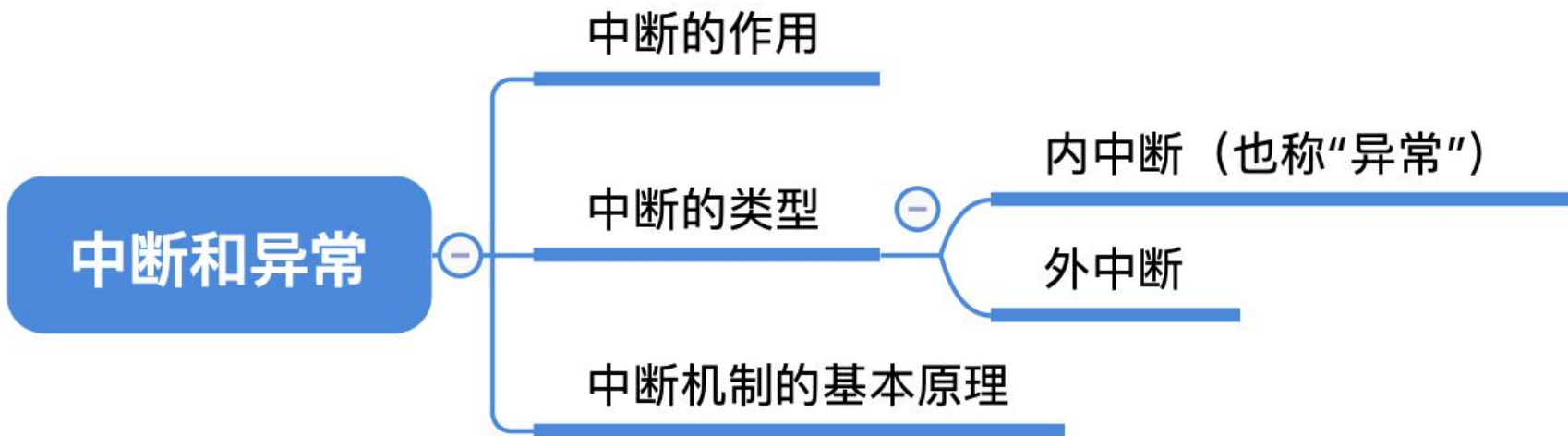
只把最基本的功能保留在内核

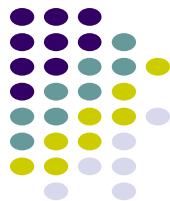
微内核

优点：内核功能少，结构清晰，方便维护

缺点：需要频繁地在核心态和用户态之间切换，性能低

# 补充-中断和异常





# 中断的作用

“中断”会使CPU由用户态变为内核态，使操作系统重新夺回对CPU的控制权

- CPU 上会运行两种程序，一种是操作系统内核程序，一种是应用程序
- 在合适的情况下，操作系统内核会把CPU的使用权主动让给应用程序（第二章进程管理相关内容）
- “中断”是让操作系统内核夺回CPU使用权的唯一途径
- 如果没有“中断”机制，那么一旦应用程序上CPU运行，CPU就会一直运行这个应用程序



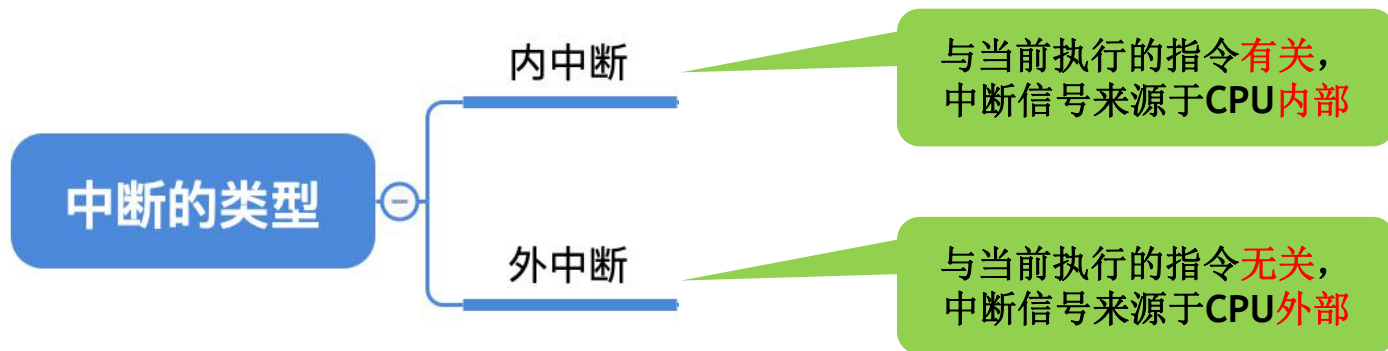
既如此，何来“并发”！？





# 中断的类型

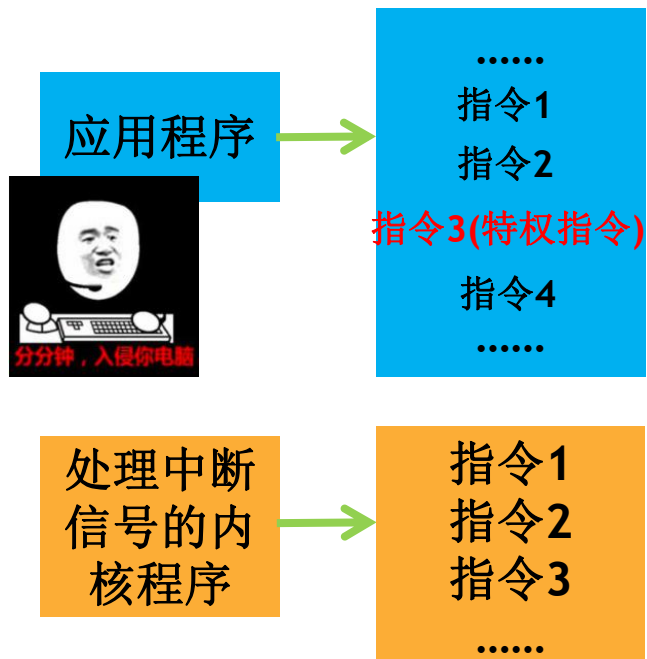
- **内核态→用户态**：执行一条**特权指令**——修改PSW的标志位为“**用户态**”，这个动作意味着操作系统将主动让出CPU使用权
- **用户态→内核态**：由“**中断**”引发，**硬件自动完成变态过程**，触发中断信号意味着操作系统将**强行夺回CPU的使用权**





# 内中断-异常

与当前执行的指令**有关**，  
中断信号来源于**CPU内部**



内核态 用户态



中断  
信号



例子 1: 试图在用户态下执行特权指令

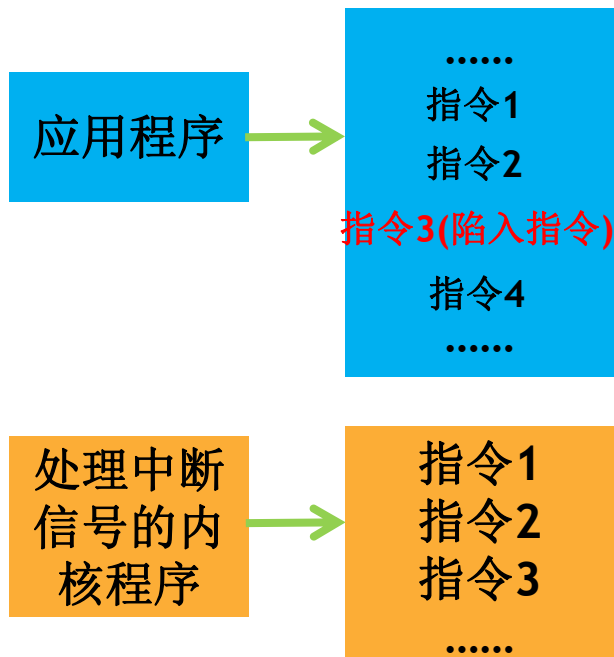
例子 2: 执行除法指令时发现除数为 0

若当前执行的指令是非法的，  
则会引发一个中断信号



# 内中断-异常

与当前执行的指令**有关**，  
中断信号来源于**CPU内部**



例子 3：有时候应用程序想请求操作系统内核的服务，此时会执行一条特殊的指令——**陷入指令**，该指令会引发一个内部中断信号

执行“陷入指令”，意味着应用程序**主动**地将**CPU**控制权还给操作系统内核。“系统调用”就是通过陷入指令完成的

# 外中断

与当前执行的指令**无关**，  
中断信号来源于**CPU外部**

应用程序1

指令1  
指令2  
指令3  
指令4  
.....

处理中断  
信号的内  
核程序

指令11  
指令22  
指令33  
.....

应用程序2

指令1  
指令2  
指令3  
.....

内核态 用户态



中断信号  
中断信号



时钟部件

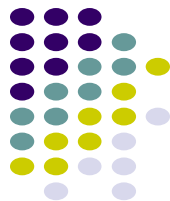
操作系统内核决定  
接下来让另一个应  
用程序上CPU运行

处理I/O中  
断的内核  
程序



当输入输出任务完成时，  
向CPU发送中断信号

时钟部件每隔一个时间片  
(如 30ms) 会给CPU  
发送一个时钟中断信号



# 中断的分类

与当前执行的指令**有关**，  
中断信号来源于**CPU内部**

## 中断的分类

内中断（也称异常、例外）



陷阱、陷入 (trap)

故障 (fault)

终止 (abort)

由陷入指令引发，是应用程序故意引发的

由错误条件引起的，可能被内核程序修复。内核程序修复故障后会**把CPU使用权**还给应用程序，让它继续执行下去。如：缺页故障。

外中断（也称“中断”）

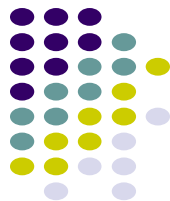


时钟中断

I/O中断请求

与当前执行的指令**无关**，  
中断信号来源于**CPU外部**

由致命错误引起，内核程序无法修复该错误，因此一般不再将**CPU的使用权**还给引发终止的应用程序，而是直接终止该应用程序。如：整数除0、非法使用特权指令



# 中断机制的基本原理

不同的中断信号，需要用不同的中断处理程序来处理。当CPU检测到中断信号后，会根据中断信号的类型去查询“中断向量表”，以此来找到相应的中断处理程序在内存中的存放位置。

中断信号类型	中断处理程序指针
0	
1	
2	
3	.....
4	.....
5	.....
.....	.....

0中断处理  
程序

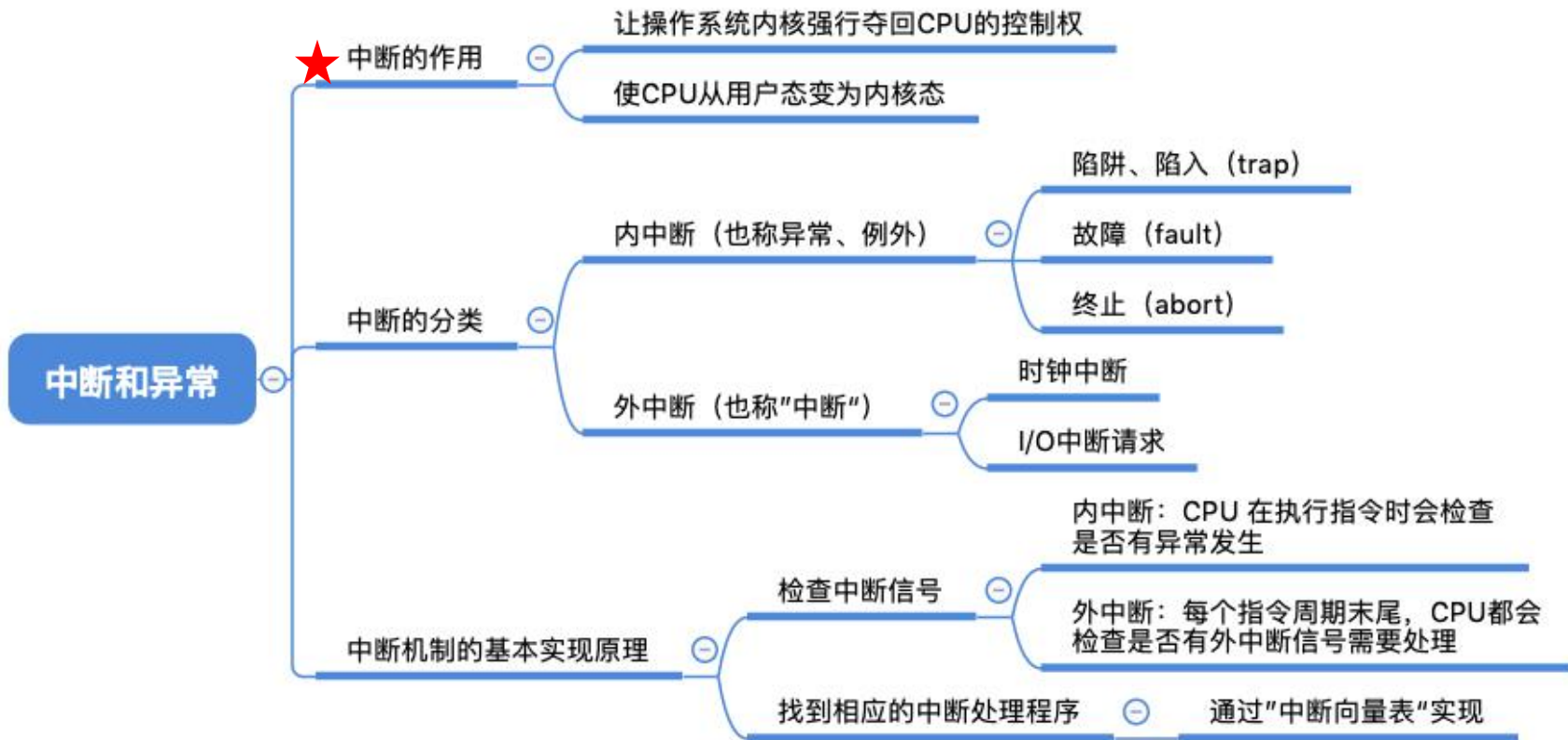
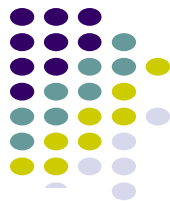
1中断处理  
程序

2中断处理  
程序

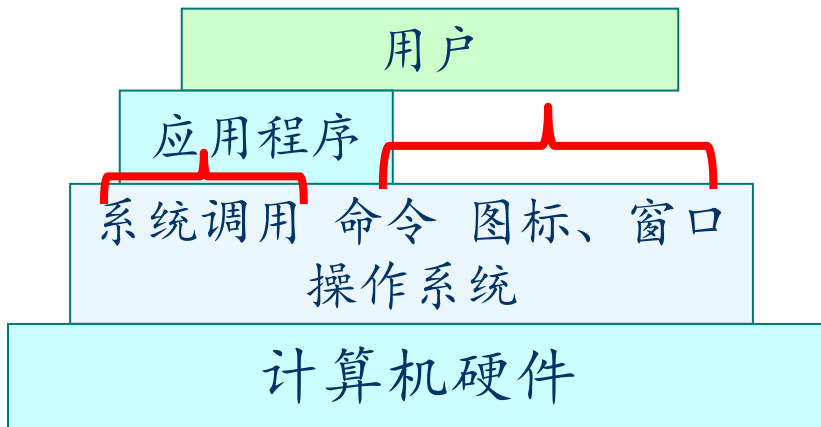
显然，中断处理程序一定是内核程序，需要运行在“内核态”

# 知识回顾

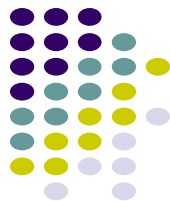
由陷入指令引发，是应用程序故意引发的



# 补充-OS接口







# 联机用户接口

- 适用：几乎所有计算机的操作系统中。
- 组成：命令+终端处理程序+命令解释程序
- 过程：用户在键盘上输入命令；
  - 终端处理程序接收命令并显示在屏幕上；
  - 命令解释程序解释并执行该命令。
- 联机命令举例：
  - LINUX或UNIX：login；logout；
  - DOS：copy；format；
- 命令解释程序：操作系统的最高层
  - MS-DOS：COMMAND.COM
  - UNIX：Shell



# 脱机用户接口

- 适用：批处理系统。
- 组成：JCL+作业说明书+命令解释程序
- 过程：用户把对作业的控制用JCL写在作业说明书上，命令解释程序按照作业说明书解释并执行。

作业控制语言



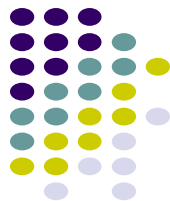
# 程序接口——系统调用

- “**系统调用**”是操作系统提供给应用程序（程序员/编程人员）使用的接口，可以理解为一种可供应用程序调用的特殊函数，**应用程序可以通过系统调用来请求获得操作系统内核的服务**

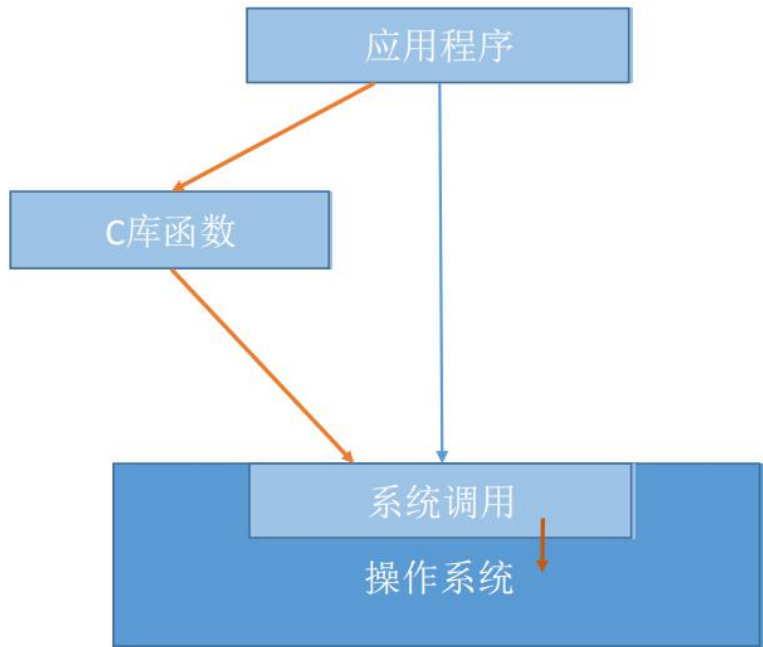
目的：为用户程序访问系统资源而设置。

组成：一组系统调用

系统调用：一个系统调用是一个能完成特定功能的子程序



# 系统调用与库函数的区别



普通应用程序	可直接进行系统调用，也可使用库函数。有的库函数涉及系统调用，有的不涉及
编程语言	向上提供库函数。有时会将系统调用封装成库函数，以隐藏系统调用的一些细节，使程序员编程更加方便。
操作系统	向上提供系统调用，使得上层程序能请求内核的服务
裸机	



**不涉及**系统调用的库函数：如的“**取绝对值**”的函数

**涉及**系统调用的库函数：如“**创建一个新文件**”的函数



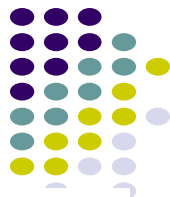
## 小例子：为什么系统调用是必须的？

生活场景：去学校打印店打印论文，你按下了 **WPS** 的“打印”选项，打印机开始工作。你的论文打印到一半时，另一位同学按下了 **Word** 的“打印”按钮，开始打印他自己的论文。

思考：如果两个进程可以随意地、并发地共享打印机资源，会发生什么？

两个进程并发运行，打印机设备交替地收到 **WPS** 和 **Word** 两个进程发来的打印请求，结果两篇论文的内容混杂在一起了...

解决方法：由操作系统内核对共享资源进行统一的管理，并向上提供“**系统调用**”，用户进程想要使用打印机这种共享资源，只能通过系统调用向操作系统内核发出请求。内核会对各个请求进行协调处理。

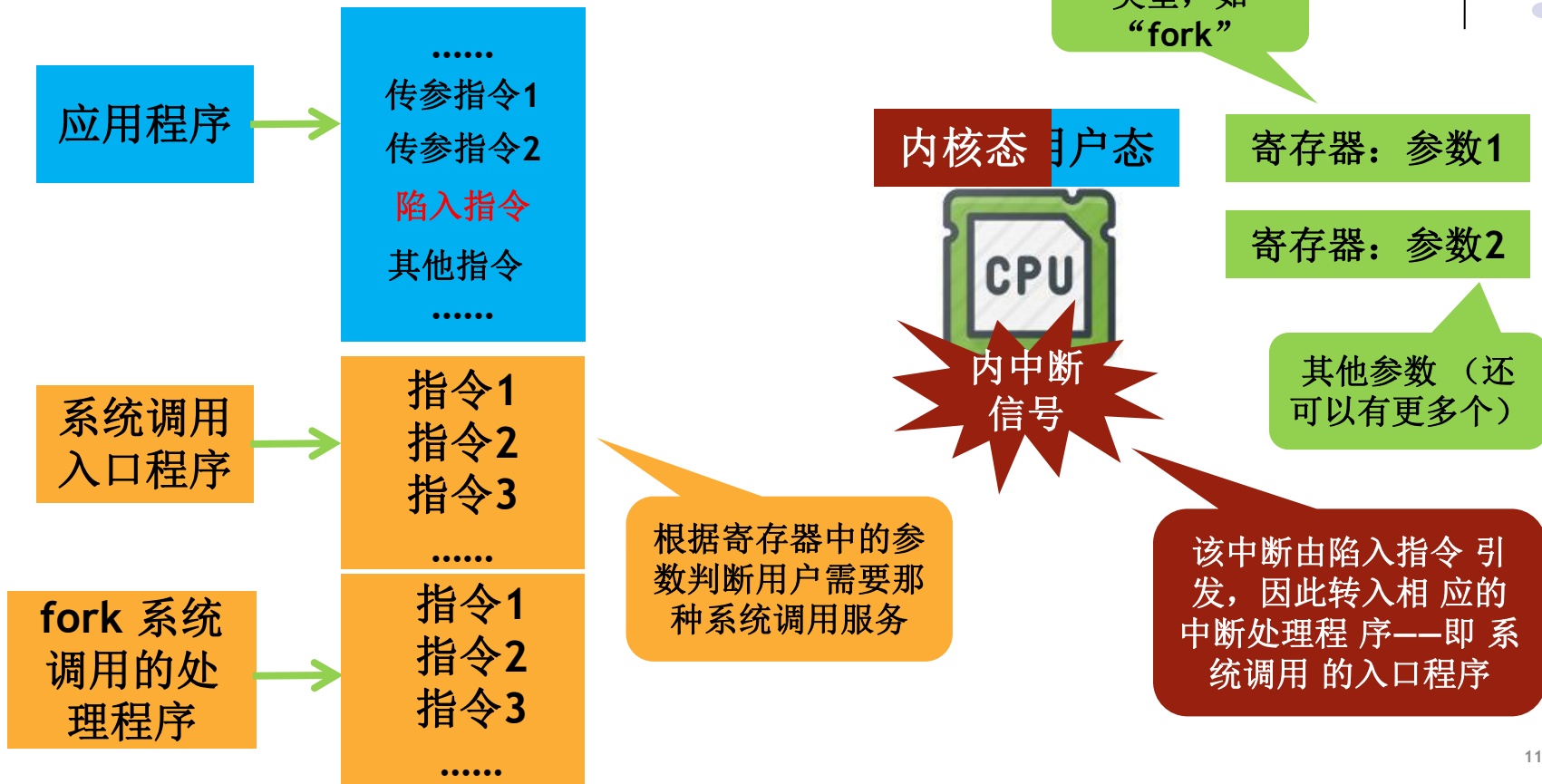


# 什么功能要用到系统调用？



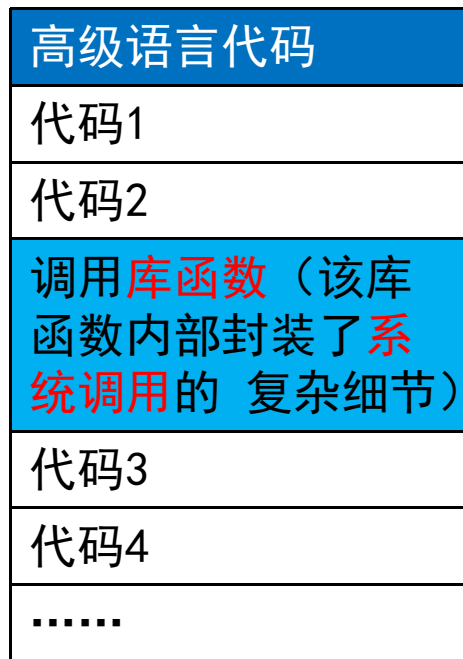
应用程序通过系统调用请求操作系统的服务。而系统中的各种共享资源都由操作系统内核统一掌管，因此凡是与共享资源有关的操作（如存储分配、I/O操作、文件管理等），都必须通过系统调用的方式向操作系统内核提出服务请求，由操作系统内核代为完成。这样可以保证系统的稳定性和安全性，防止用户进行非法操作。

# 系统调用的过程

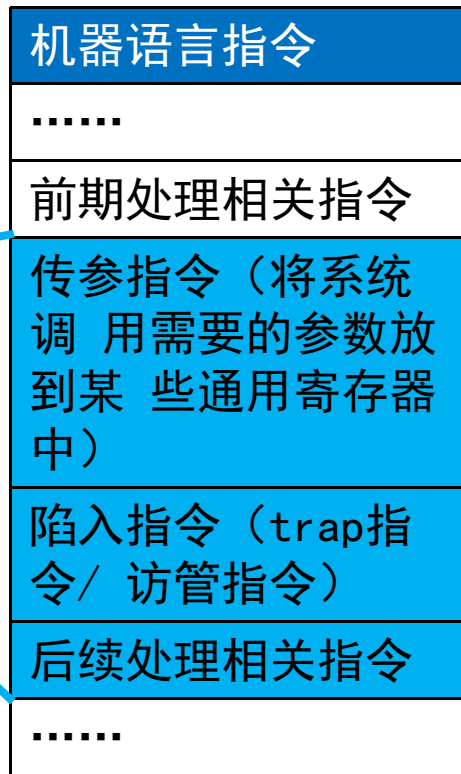




# 系统调用的过程

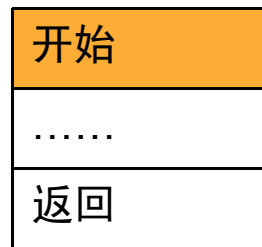


应用程序（机器语言视角）  
运行在用户态



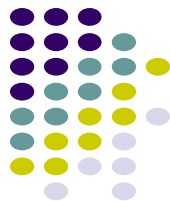
应用程序（机器语言视角）  
运行在用户态

陷入指令=trap指令=访管指令



处理系统调用的  
内核程序（运行在核心态）

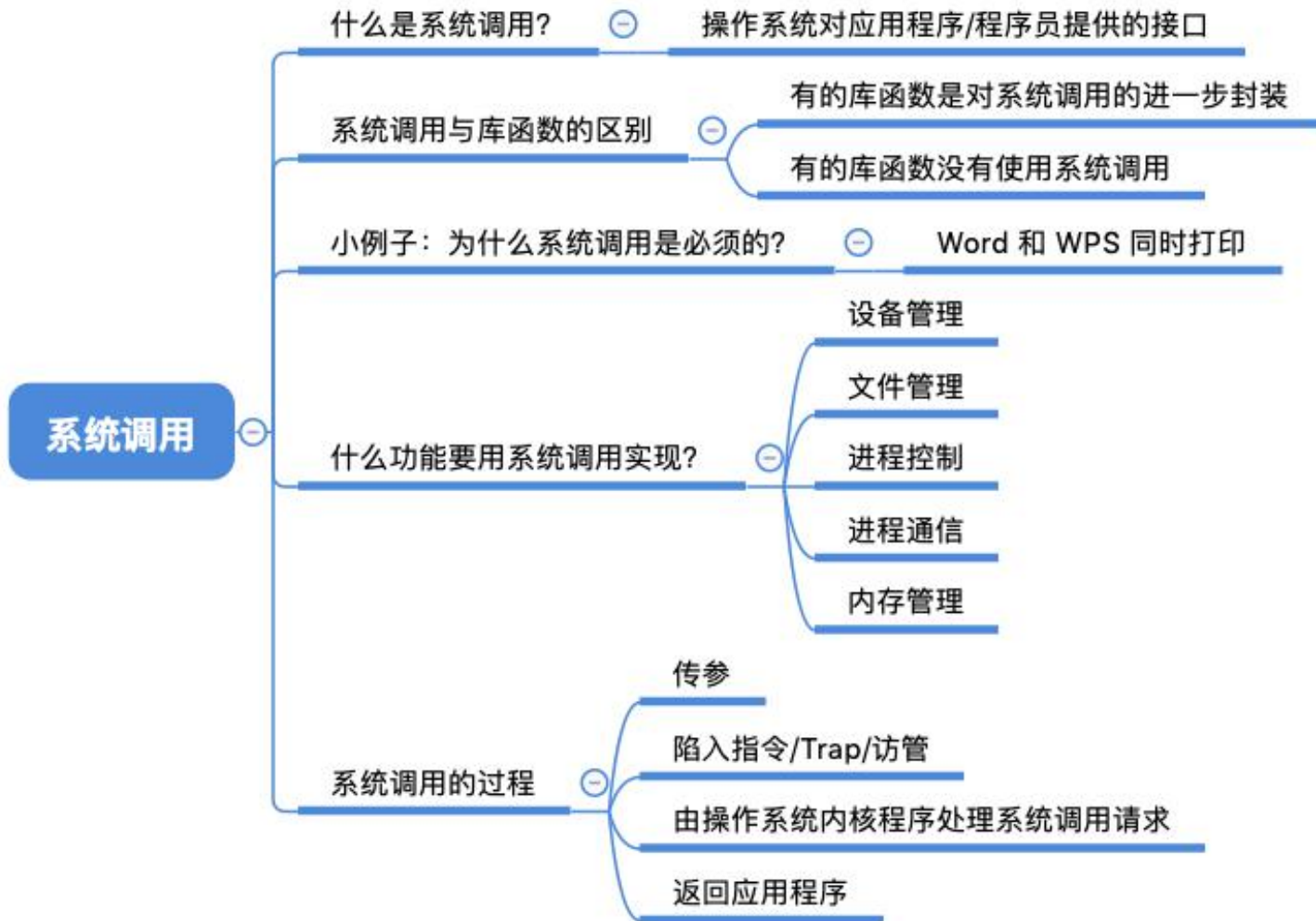




# 系统调用的过程

- 传递系统调用参数 → 执行陷入指令（**用户态**） → 执行相应的内请求核程序处理系统调用（**核心态**） → 返回
- 注意：
  - 1. **陷入指令**是在**用户态**执行的，执行陷入指令之后立即引发一个**内中断**，使CPU进入核心态
  - 2. **发出系统调用请求**是在**用户态**，而**对系统调用的相应处理**在**核心态**下进行

# 知识回顾





# 本章总结

- 操作系统定义
- 设计目标：方便性、有效性、可扩充性、开放性
- 3个作用：用户与硬件间接口、资源管理者、资源抽象
- N种基本类型：单道、多道、分时、实时
- 操作系统的4个特征：并发、共享、虚拟、异步
- 4个主要功能：处理器管理、存储器管理、设备管理、文件管理
- 操作系统体系结构与内核：运行机制、内核、中断、接口（系统调用）

# 作业



- 操作系统的作用是什么？
- 操作系统具有哪几个特征？它们之间有何关系？
- 试述多道程序设计技术的基本思想。为什么采用多道程序设计技术可以提高资源利用率？
- 什么是分时系统？其主要特征是什么？适用于哪些应用？