第一章 操作系统概论

上堂课总结：

1. 计算机系统简介
2. 硬件与软件简介
3. 什么是操作系统
4. 操作系统发展历史

操作系统核心技术

- 资源复用技术
  - 解决资源不足问题
- 资源虚拟化
  - 解决资源不足问题
  - 与抽象技术相结合，向用户屏蔽系统的复杂性
- 资源抽象
  - 向用户屏蔽系统的复杂性，解决系统的易用性问题

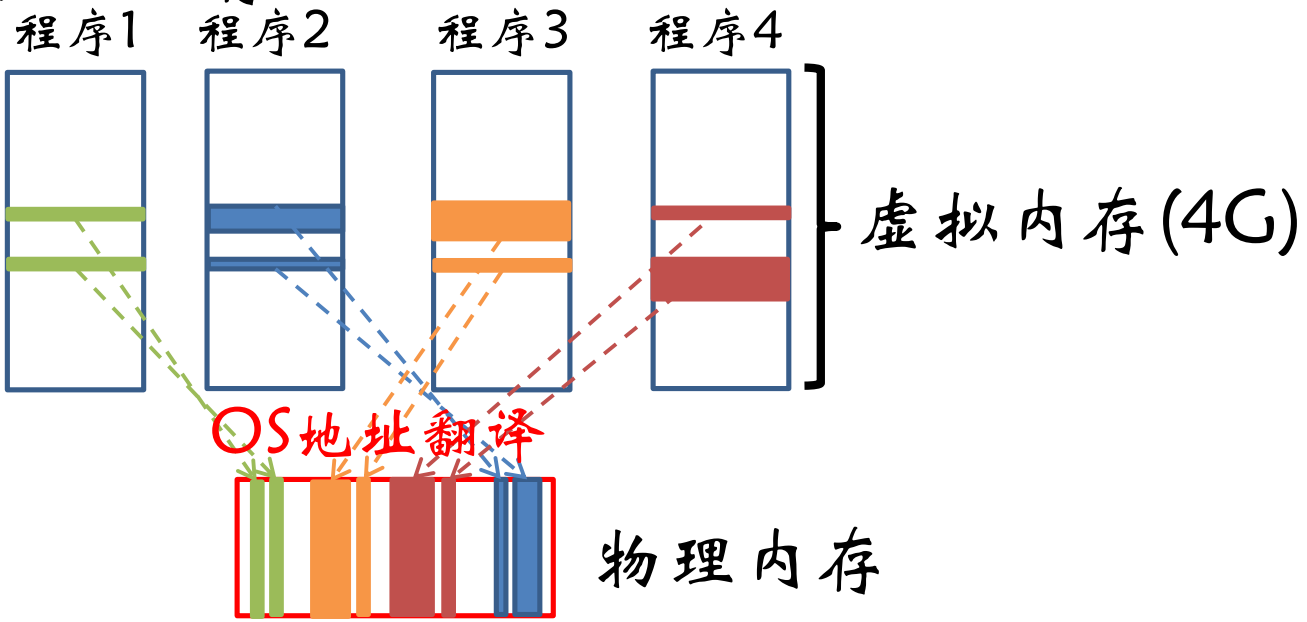
资源复用

- **空分复用**：将资源划分成小块然后分配给不同的进程
  - 例：主存的空分复用

- **时分复用**：将时间划分成片，然后分配给进程，进程可以在一个时间片内独享资源
  - **时分独占式**——进程获得时分独占式资源后，对资源执行多个操作，通常使用一个完整的周期后才会释放(如磁带机)
  - **时分共享式**——时分共享式资源指进程占用该类资源使用后，很可能随时被剥夺，被另一个进程抢占使用(如处理器)

资源虚拟化

- 每一个应用程序都亲自考虑如何复用资源？
- 资源虚拟化：
  - 单个物理资源→逻辑上的多个对应物
  - 假象：进程独占资源
  - 由OS负责完成底层物理资源的复用
- 例：虚拟内存

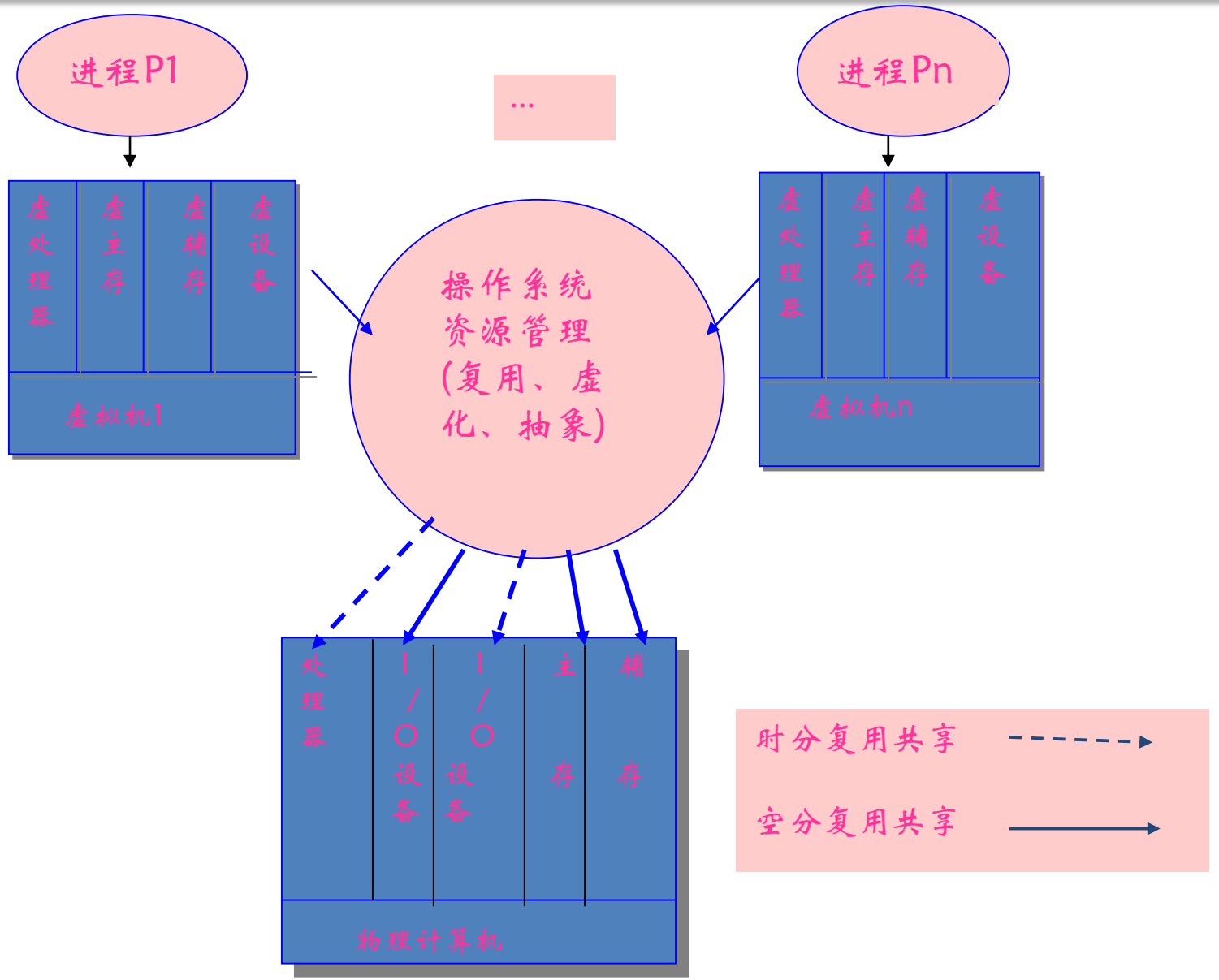


资源抽象

- 通常与资源虚拟化组合使用
- 创建软件来向用户屏蔽资源的物理特性和接口细节，简化对硬件的访问和操作
  - 内部：使用资源复用技术，实现复杂的资源管理（分配，访问等）
  - 外部：提供应用接口(API)
- 例： 内存分配函数
  - void* malloc(unsigned size);
  - 在主存哪里分配？会不会和其他程序冲突？内存满了怎么办？

应用程序完全不需要知道！一切交给操作系统！！！

资源管理技术小结


进程P1
...
进程Pn
虚处理器
虚主存
虚辅存
虚设备
虚拟机1
操作系统资源管理(复用、虚化、抽象)
虚处理器
虚主存
虚辅存
虚设备
虚拟机n
处理器
I/O设备
I/O设备
主存
辅存
物理计算机
时分复用共享
空分复用共享

操作系统中最基础的抽象

- 进程
  - 是对已进入主存正在运行的程序在处理器上操作的状态集的抽象
- 虚存
  - 是对物理主存的抽象，进程可获得一个硕大的连续地址空间来存放可执行程序和数据，可使用虚拟地址来引用物理主存单元
- 文件
  - 是对磁盘之类存储设备的抽象。

操作系统的基础抽象——进程抽象

进程（process）

（进程是处理器的一种抽象）

用户：运行应用程序，以进程方式执行

—虚拟机界面—fork()、wait()、exec()……

OS：进程及其管理

—物理机界面—进程调度和上下文切换

硬件：处理器

进程是对于进入内存的执行程序在处理器上操作的状态集的一个抽象。进程抽象的效果是让用户感觉到有自己独享的处理器，从而，可为用户提供多任务操作系统和分时操作系统。

进程是并发和并行操作的基础。概念上每个进程都是一个自治执行单元；实际上是透明地共享一个或多个处理器。

操作系统的基础抽象——虚存抽象

虚 存(virtual memory)

（虚存是内存的一种抽象）

用户：运行应用程序，使用逻辑地址

—虚拟机界面—虚拟地址

OS:虚存及其管理

—物理机界面—物理地址

硬件:主存+辅存

虚存抽象的效果是给用户造成假象，感觉独占了一个连续地址空间，编写应用程序的长度不受物理内存大小限制。

虚存是通过结合对内存和外存的管理来实现的，把一个进程的虚存中的内容存储在磁盘上，用内存作为磁盘的高速缓存，以此为用户提供比物理内存空间大得多的虚拟内存空间。

操作系统的基础抽象——文件抽象

文件(file)

（文件是设备的一种抽象）

用户：运行应用程序，使用文件

—虚拟机界面—open()、read()、write()...

OS:文件及其管理

—物理机界面—设备驱动

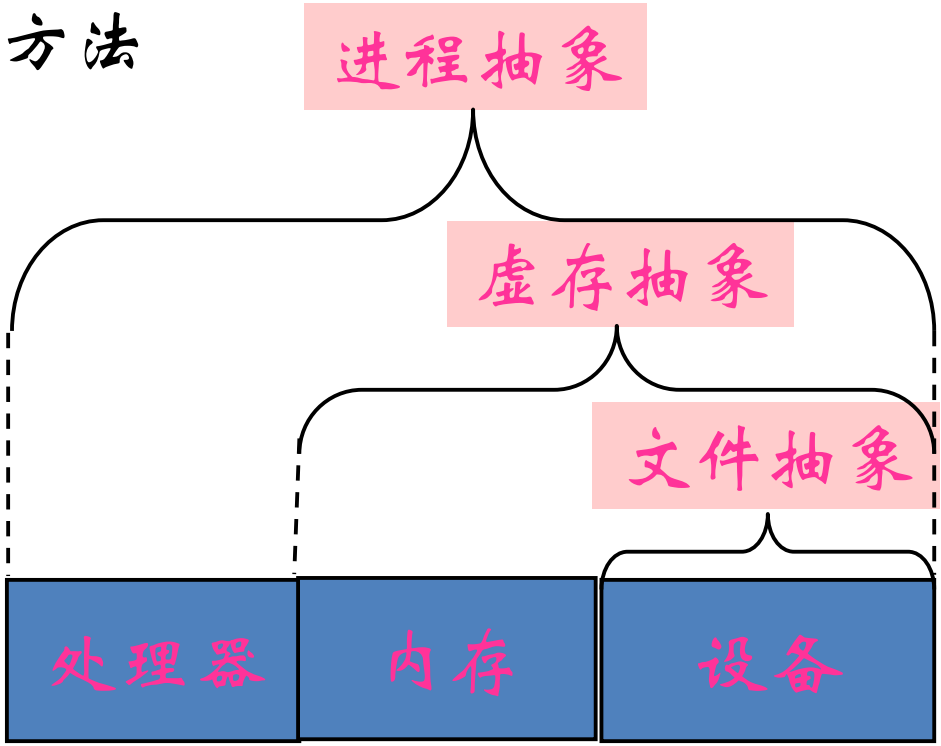
硬件:磁盘及其他设备

文件是通过将文件中的字节映射到存储设备的物理块中来实现文件抽象。

文件抽象的效果是让用户感觉到总能满足自己对设备上信息的存取需求，而且使用十分方便。

基础抽象的包含关系

- 操作系统基本任务
  - 防止硬件资源被失控的应用程序滥用
  - 屏蔽复杂的硬件操作细节，为应用程序提供使用硬件资源的简单且一致的方法
- 抽象目标
  - 建立多层抽象



其他资源抽象

- 低层硬件资源抽象，提供对外使用的抽象接口
  - 中断、时钟、网络接口等
- 没有特定基础硬件的软件资源
  - 消息、信号量、共享数据结构

操作系统主要特性

- 并发性（concurrency）
- 共享性（sharing）
- 异步性（asynchronism）
  - 也称随机性

操作系统主要特性——并发性

- 指两个或两个以上的事件或活动在同一时间间隔内发生
  - 发挥并发性能够消除系统中部件和部件之间的相互等待
  - 有效地改善系统资源利用率，改进系统吞吐率，提高系统效率

操作系统主要特性—并发性

- 并发性使系统变得复杂化
  - 如何从一个活动切换到另一个活动
  - 怎样将各个活动隔离开来，使之互不干扰，免遭对方破坏
  - 怎样让多个活动协作完成任务
  - 怎样协调多个活动对资源的竞争
  - 如何保证每个活动的资源不被其它进程侵犯
  - 多个活动共享文件数据时，如何保证数据的一致性

操作系统主要特性——并发性

- 采用并发技术的系统称多任务系统
  - 并发的实质是一个物理CPU(也可以多个物理CPU)在若干道程序之间多路复用
  - 并发性是对有限物理资源强制行使多用户共享以提高效率
- 实现并发技术的关键之一是如何对系统内的多个活动(进程)进行切换

操作系统主要特性——并发性

- 并行性（Parallelism）
  - 指两个或两个以上的事件或活动在同一时刻发生
  - 在多道程序环境下，并行性使得多个程序同一时刻可以在不同的CPU上执行
- 并行性 *VS.* 并发性
  - 并行的事件或活动一定是并发的，但反之并发的事件或活动未必是并行的
  - 并行性是并发性的特例，而并发性是并行性的扩展

操作系统主要特性——共享性

- 指操作系统中的资源可被多个并发执行的进程所使用
  - 透明资源共享：资源隔离与授权访问
  - 独占资源共享：临界资源与独占访问
- 与共享性有关的问题
  - 资源分配
  - 信息保护
  - 存取控制
  - ……

共享性 VS. 并发性

- 共享性和并发性是操作系统的两个基本特性，互为依存
  - 一方面，资源共享是由程序的并发执行而引起的，若系统不允许程序并发执行，也就不存在资源共享问题
  - 另一方面，资源共享是支持并发性的基础，若系统不能对资源共享实施有效管理，必然会影响程序的并发执行，甚至导致程序无法并发执行

操作系统主要特性——异步性

- 操作系统中的异步性处处可见
  - 进程“何时执行、何时暂停、何种速度向前推进”都是异步(随机)的
  - 作业到达系统的类型和时间是随机的
  - 操作员发出命令或按按钮的时刻是随机的
  - 程序运行发生错误或异常的时刻是随机的
  - 各种各样硬件和软件中断事件发生的时刻是随机的

操作系统主要特性——异步性

- 多个程序并发执行，并发活动会导致随机事件的发生
  - 并发程序是以异步方式运行的，系统的功能及服务因响应事件而被激活，但是事件以不均等的频率发生
- 异步性给系统带来潜在危险，有可能导致与时间有关的错误，操作系统的一个重要任务是
  - 确保捕捉任何一种随机事件
  - 正确处理可能发生的随机事件
  - 正确处理任何一种产生的事件序列
- 否则将会导致严重后果

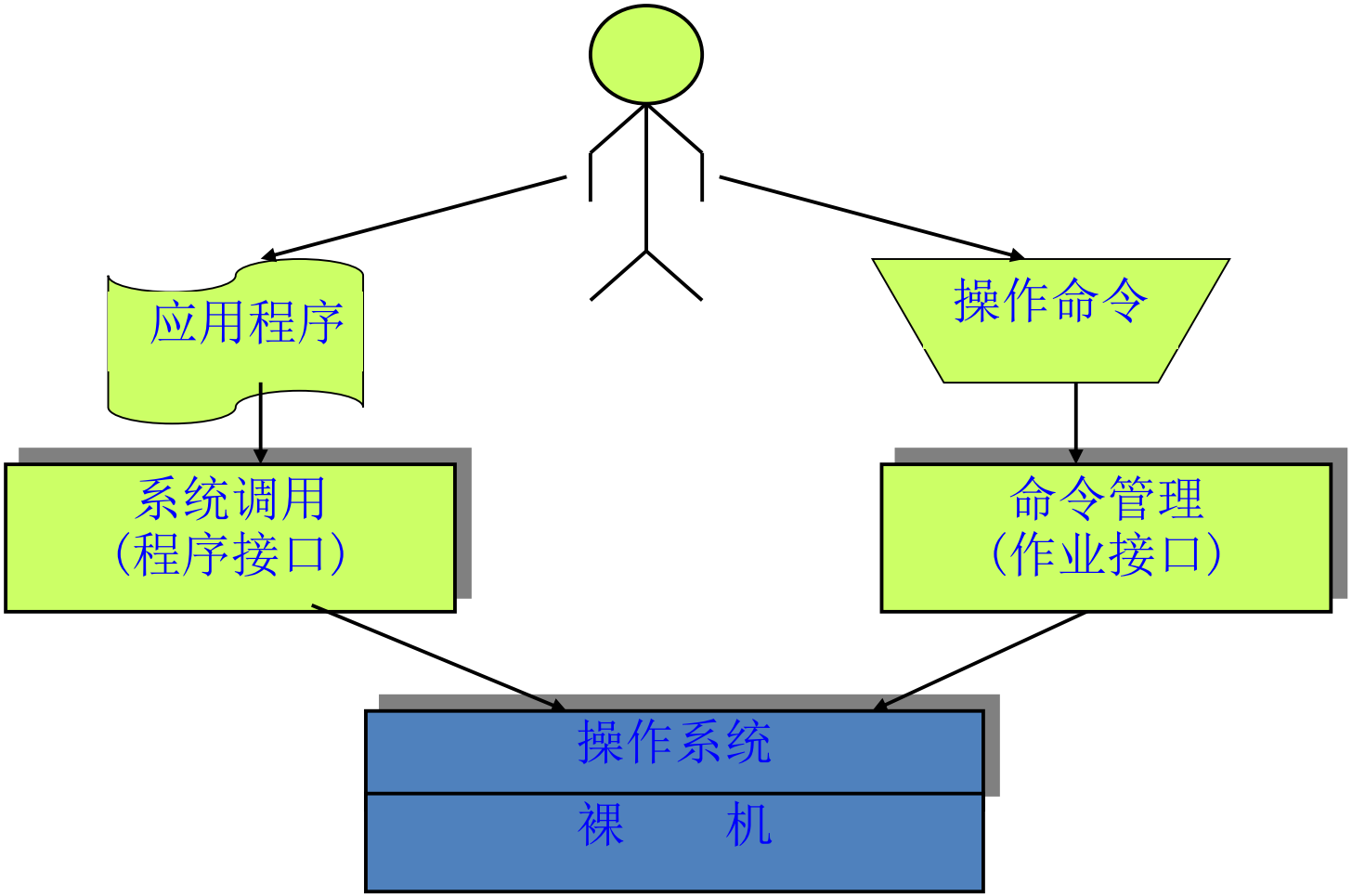
操作系统提供的服务与接口

- 1.3.1 基本服务
- 1.3.2 程序接口与系统调用
- 1.3.3作业接口与操作命令

操作系统提供的基本服务

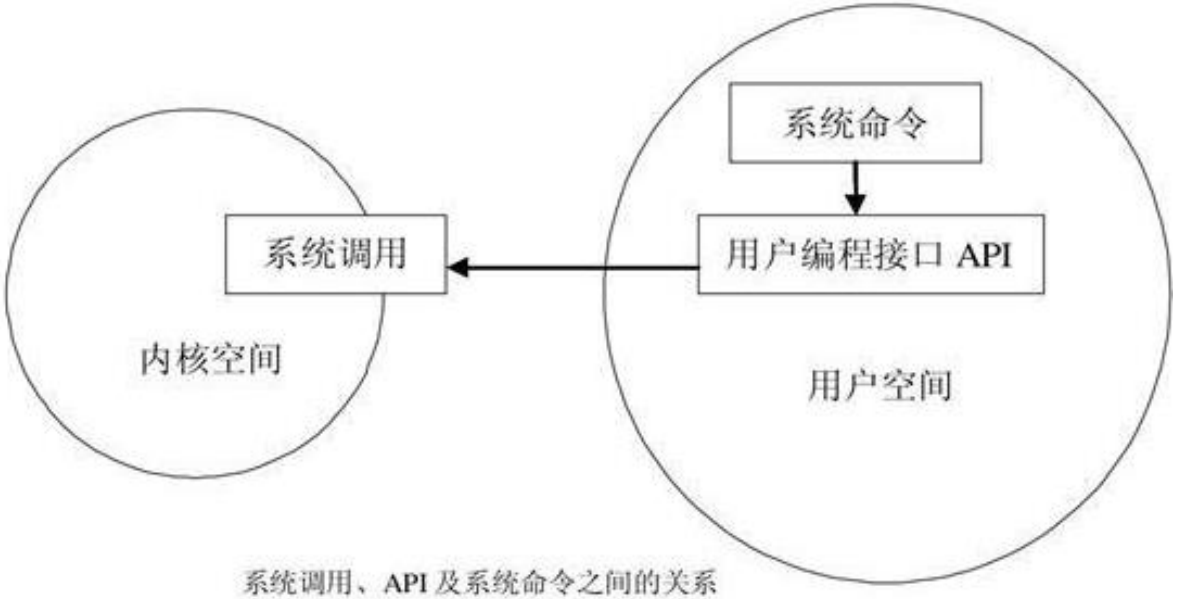
- 创建程序
- 执行程序
- 数据I/O
- 信息存取
- 通信服务
- 错误检测和处理
- 还具有另外一些功能:资源分配,统计,保护。

操作系统提供的接口



操作系统提供的程序接口(1)

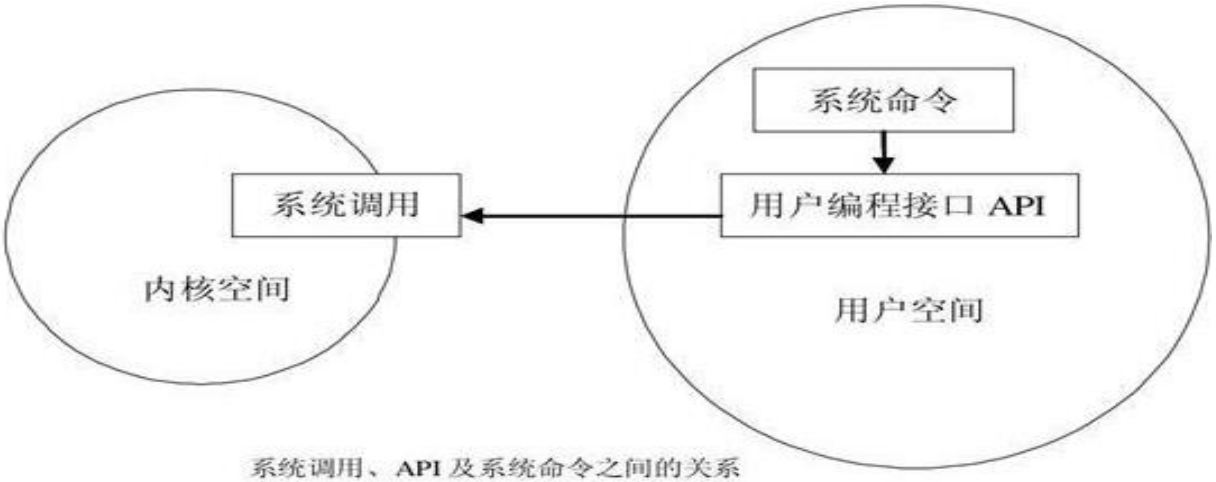
- 什么是系统调用(System Call)?
  - 操作系统提供给用户程序调用的一组“特殊”接口(用户程序与内核的中介)
  - 用户程序可以通过这组“特殊”接口来获得操作系统内核提供的服务



系统调用、API 及系统命令之间的关系

操作系统提供的程序接口 (2)

- 系统调用的作用
  - 封装抽象资源，方便用户使用
    - 内核服务很复杂，但接口相对简单
  - 保护系统安全
    - 用户程序访问内核的唯一合法途径
    - 内核可基于权限和规则对资源访问进行裁决，保证系统的安全性



系统调用、API 及系统命令之间的关系

操作系统提供的程序接口 (3)

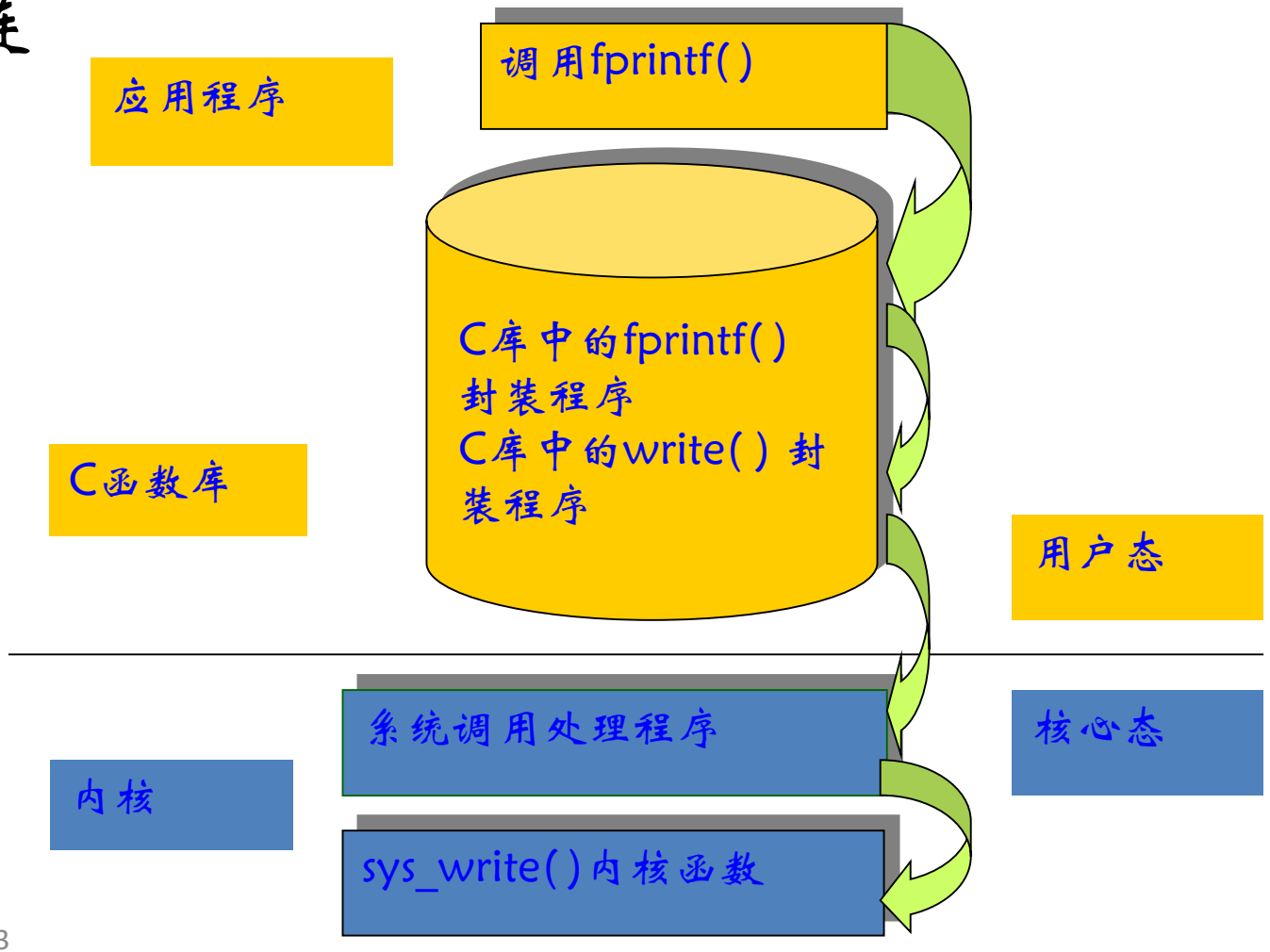
- 系统调用的再抽象
  - 系统调用接口仍然不够简单
  - 无法跨平台
  - POSIX(Portable Operating System Interface for Computer Environment)标准

- 语言函数库(API)： glibc、libc
  - 对系统调用的再封装
  - 高级语言


```
int read(int fd,char *buf,int n)
{
    long __res;
    __asm__ volatile(
        "int $0x80"
        :"=a" (__res)
        :"0" (__NR_read),"b" ((long)(fd),"c" ((long)(buf),"d" ((long)(n)));
        if(__res>=0)
            return int __res;
        errno=-__res;
        return -1;
}
```

操作系统提供的程序接口 (4)

- 应用程序、库函数、系统调用的调用关系链



操作系统提供的程序接口 (5)

- 系统调用的分类

(1)进程和作业管理；

(2)文件操作；

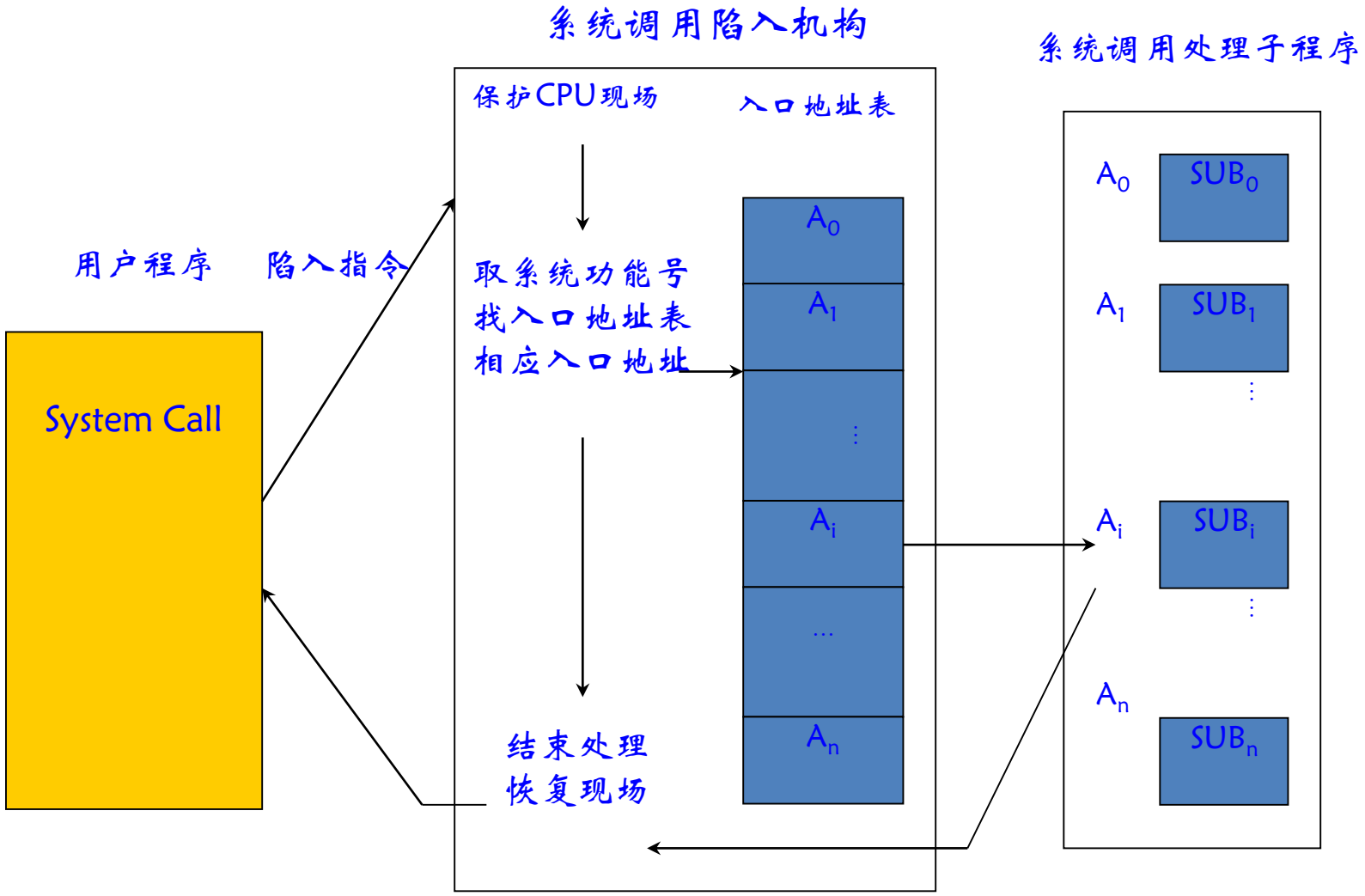
(3)设备管理；

(4)主存管理；

(5)信息维护；

(6)通信

系统调用的实现


系统调用陷入机构
系统调用处理子程序
保护CPU现场
入口地址表
用户程序
陷入指令
System Call
取系统功能号
找入口地址表
相应入口地址
结束处理
恢复现场
A0
A1
Ai
...
An
SUB0
SUB1
SUBi
SUBn

系统调用的参数传递

- 由访管指令或陷入指令自带参数，
  - 直接参数
  - 间接参数

- 通过CPU的通用寄存器传递参数，或在主存的一个块或表中存放参数，其首地址送入寄存器，实现参数传递。

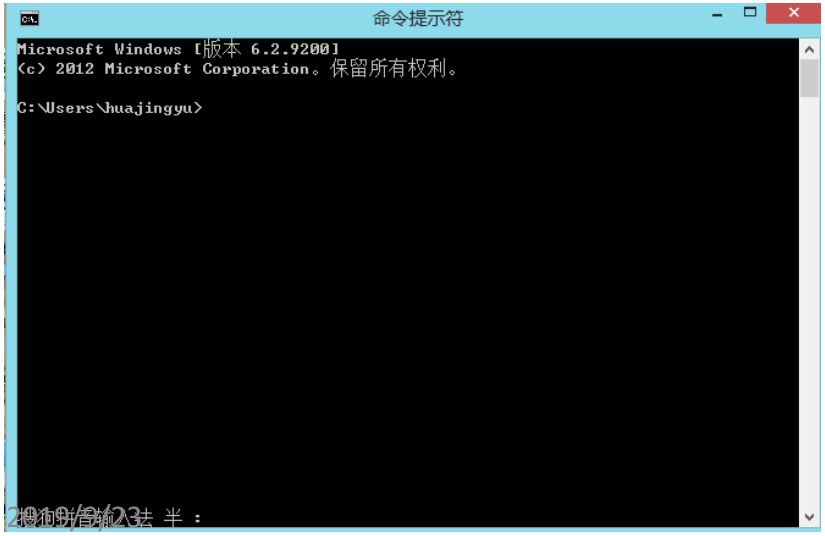
- 在主存中开辟专用堆栈区域传递参数

系统调用与函数调用的区别

- 调用形式和实现方式不同
  - 函数调用：转向的地址固定不变、在用户态执行
  - 系统调用：由功能调用号决定内核服务例程入口地址，在内核态执行
- 被调用代码的位置不同
  - 函数调用：静态调用，调用程序和被调用代码处于同一程序
  - 系统调用：动态调用，服务例程位于操作系统内核中
- 提供方式不同
  - 函数调用：编程语言提供，取决于语言提供的函数功能
  - 系统调用：由操作系统统一提供

操作系统提供的作业接口

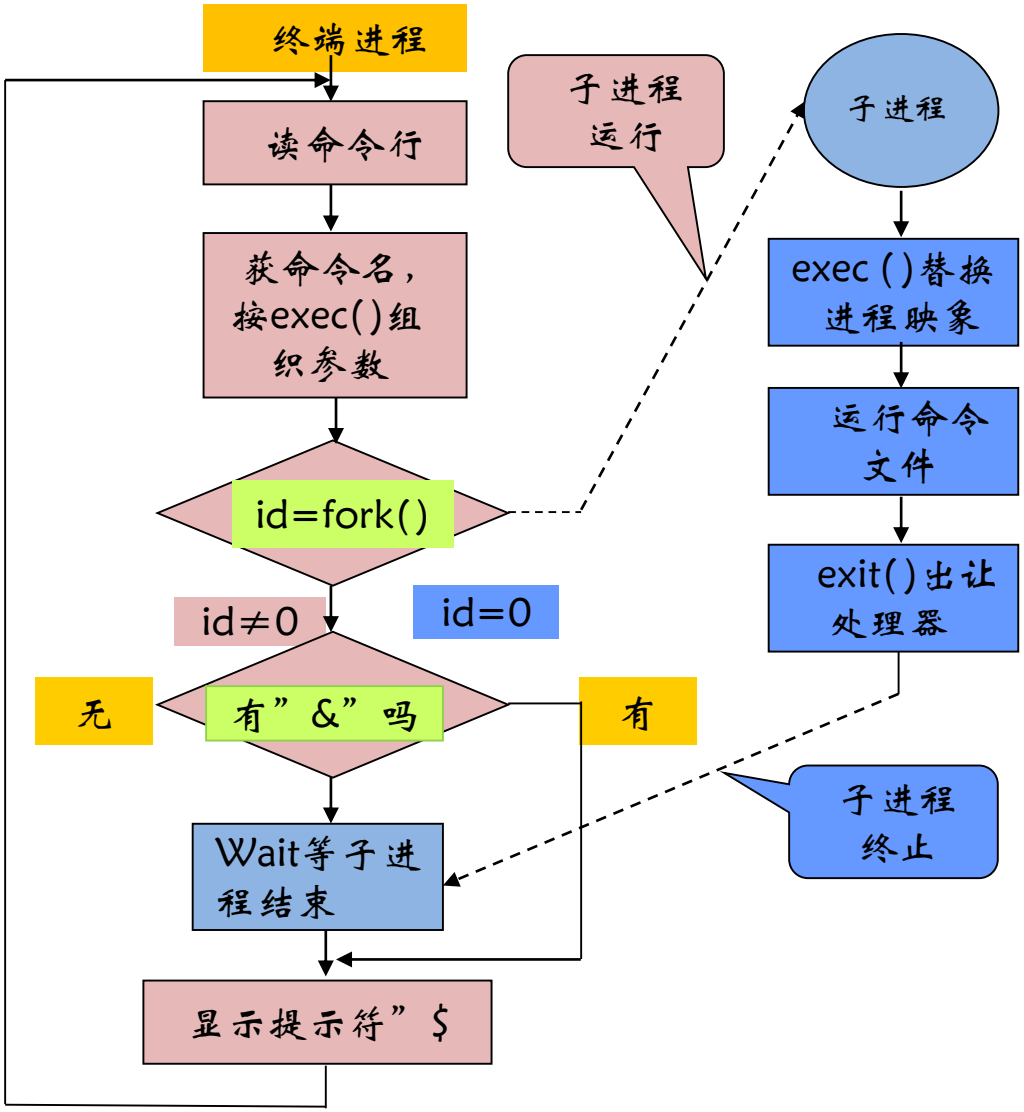
- 为用户提供的操作控制计算机工作和提供服务手段的集合
  - 命令行
  - 图形操作界面
  - 作业控制语言（批处理系统）



命令解释程序

- 命令解释程序功能
  - 接收用户输入的命令并解释执行命令
- 命令实现方式
  - 方式1：是自带命令执行代码，收到命令后，便转向相应命令处理代码执行
    - 可以使用“系统调用”帮助完成任务，由于用到终端进程的地址空间，故这类命令不宜过多
  - 方式2：是由专门“实用程序”实现，执行时把命令对应的处理文件装入主存，完成命令功能
  - 大多操作系统把两者结合起来
    - 简单命令由命令解释程序直接处理
    - 复杂命令由独立的实用程序完成

Linux命令解释器shell



1.4 操作系统构件与运行模型

- 操作系统变得越来越复杂
  - 代码量几何增长：Win2000： 3200万行，2500开发人员
  - 周期长、bug多
- 软件工程的视角研究操作系统
- 操作系统机构设计
  - 整体结构，模块划分
  - 局部结构、数据结构等
  - 运行时组织方式

操作系统构件

- 操作系统的基本单位称为构件，包括：
  - 内核
  - 进程
  - 线程
  - 类程
  - 管程

操作系统内核 (1)

- 什么是内核？
  - 操作系统的核心部分
  - 作为可信软件来提供支持进程并发执行的基本功能和基本操作的一组程序模块
  - 运行于核心态
  - 负责管理系统的进程、内存、设备驱动程序、文件和网络系统，决定着系统的性能和稳定性
- 扩展硬件
  - 安全
  - 效率

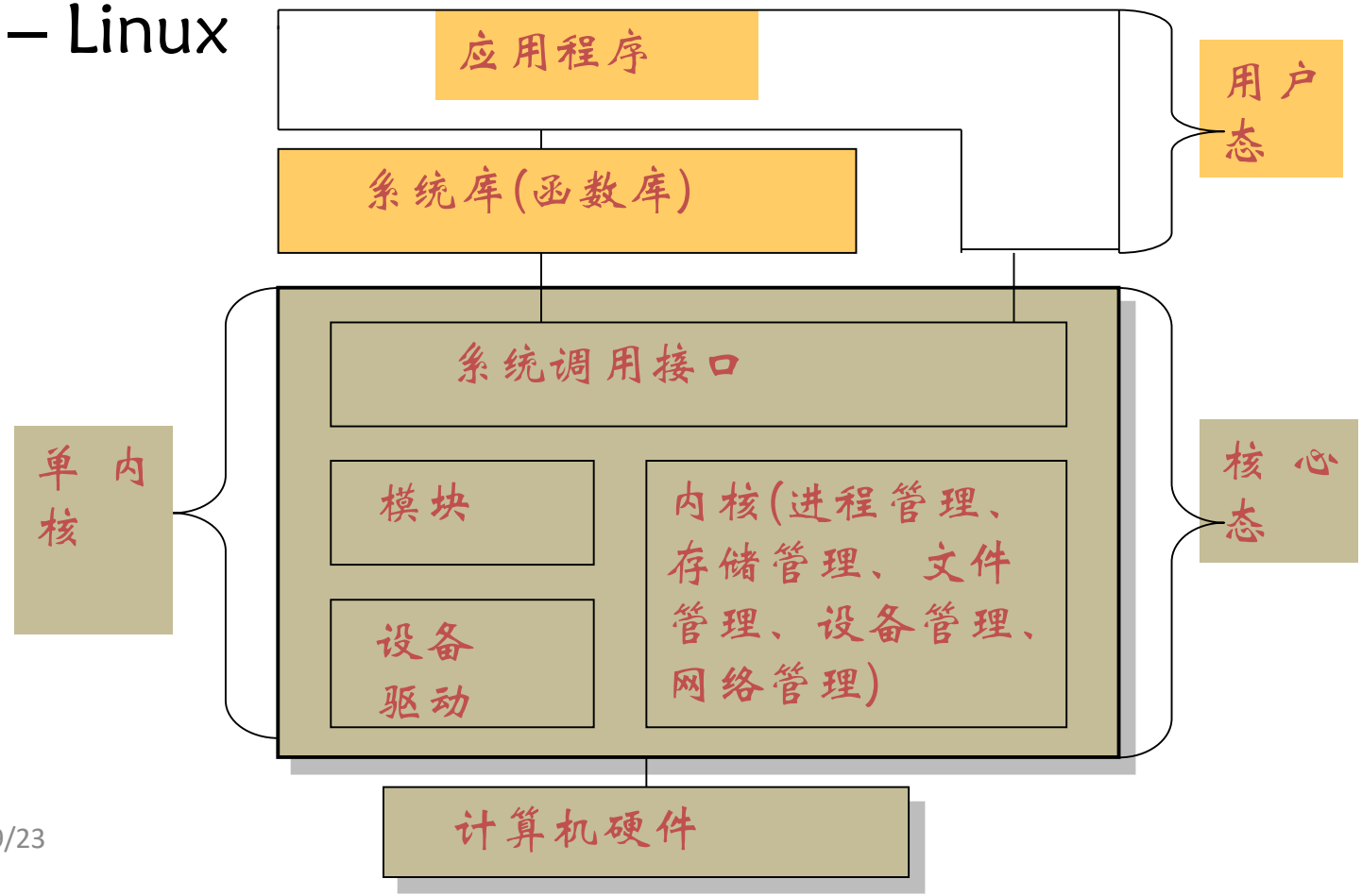
操作系统内核(2)

内核必须实现的功能：

- 中断处理
- 短程调度
  - 协调处理器竞争
- 原语(primitive)管理
  - 协调进程通信、并发执行、资源共享
  - 通信原语
  - 同步原语

操作系统内核(3)

- 单内核
  - 所有内核模块处于同一个二进制映像
  - Linux

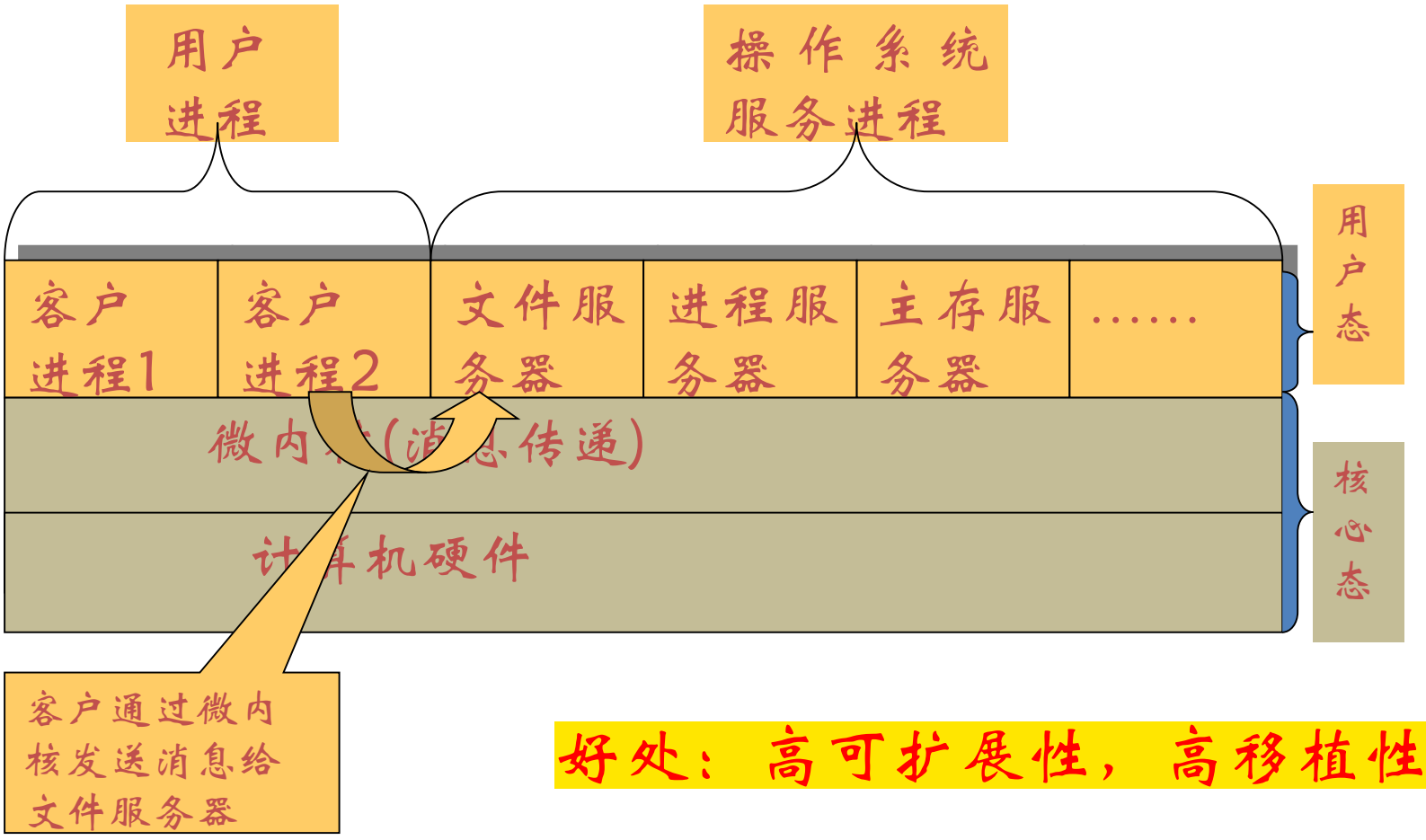


操作系统内核(4)

- 单内核两种结构
- 整体式
  - 模块间高度耦合
  - 但运行效率高
- 层次式
  - 模块划分层次，单向依赖
  - 解耦合
  - 运行效率低

操作系统内核(5)

- 微内核



好处：高可扩展性，高移植性

内核的作用

- **资源抽象**
  - 抽象硬件，方便用户程序使用
- **资源分配**
  - 将硬件资源合理分配给用户程序
- **资源共享**
  - 提供资源共享的同步、互斥机制

内核的基本属性

- 内核是由中断驱动的
- 内核是不可抢断的
- 内核部分程序在屏蔽中断状态下执行
- 内核可以使用特权指令

虚拟机的概念

- 内核向每一个用户程序虚拟了一台计算机
- 没有中断
- 独占资源
- 虚拟机为进程或模块提供了功能较强的指令系统
  - 非特权指令
  - 系统调用

内核设计思想

- 机制与策略分离
  - 内核实现有特定目的和功能的函数(机制)
  - 应用程序决定如何使用这些函数(策略)
- 例子：调度机制与调度策略的分离
- 机制与策略分离的原则：

(1)机制由OS实现，策略留给用户完成；

(2)机制放在底层，策略放在高层；

(3)机制集中在少数模块，策略拟散布在多处。

内核模块与应用程序的差别

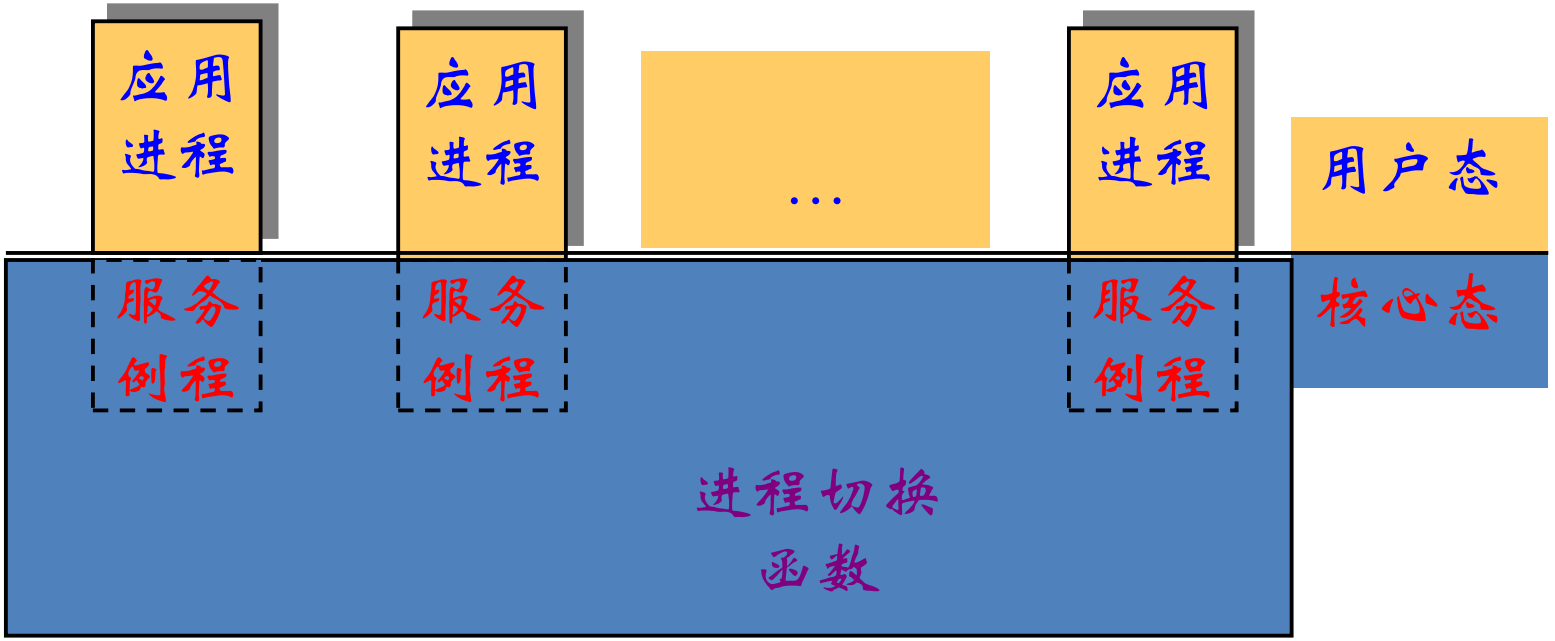
	C语言程序	模块
运行	用户空间	内核空间
入口	main()	module_init()
出口	无	module_exit()
编译	gcc -c	编制Makefile，并调用gcc
链接	gcc	insmod
运行	直接运行	insmod
调试	gdb	kdbug, kdb, kgdb等

操作系统运行模型

- 操作系统本身是一组程序，也在处理器上运行
  - 操作系统程序是否组织成进程
  - 是如何控制和怎样执行的
  - 在什么模式下运行
- 从操作系统的运行方式来看，可分成
  - 嵌入应用进程中运行模型
  - 作为独立进程运行模型

操作系统运行模型

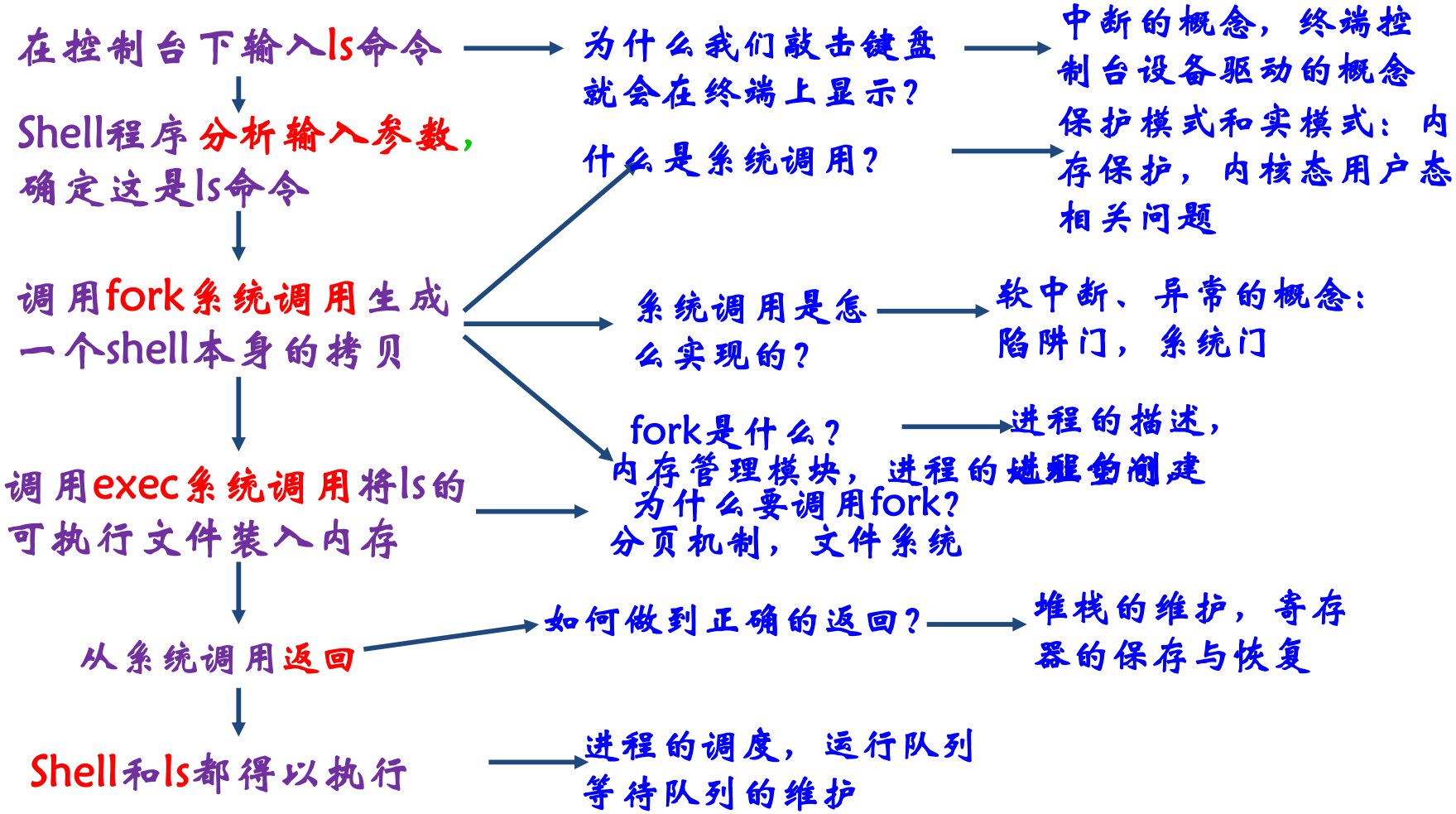
- 嵌入应用进程中运行模型
  - 作为服务例程（子程序）通过系统调用为应用进程提供服务
    - 服务例程仍然运行于当前应用进程中，但在管态执行
    - 利用应用进程的核心栈作为服务例程的工作栈



操作系统运行模型

- 作为独立进程运行模型
  - 应用进程的服务请求和服务器进程的服务响应通过微内核的消息传递机制实现

操作系统运行机制（Linux为例）



习题

- 习题一：
- 思考题：1、11
- 应用题：7
- 下下周一(9.16)上课前交(可提交电子版，微信发给我)