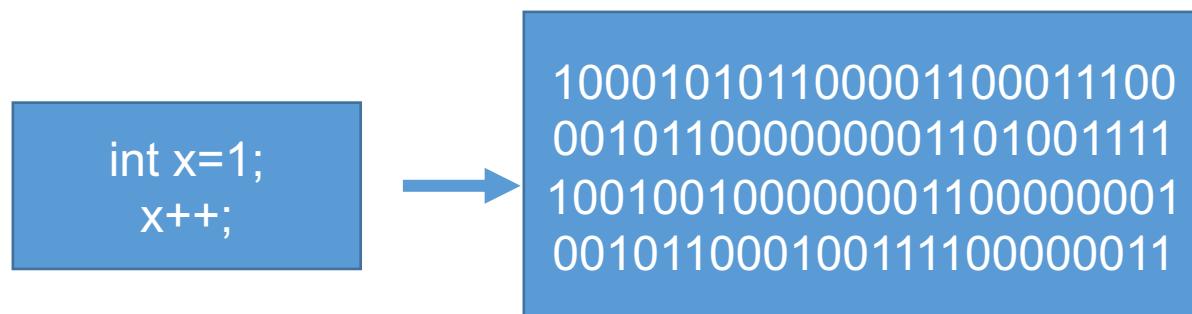
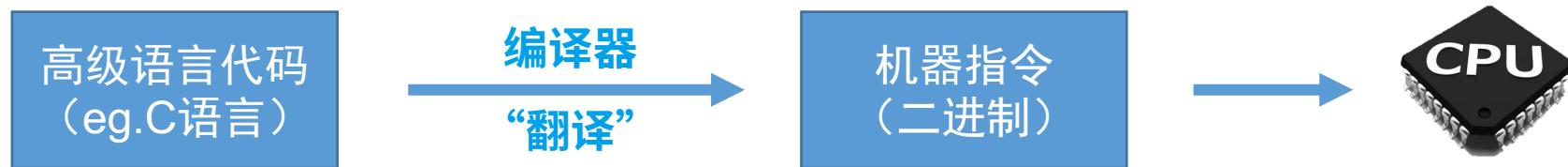


# 操作系统的根本概念

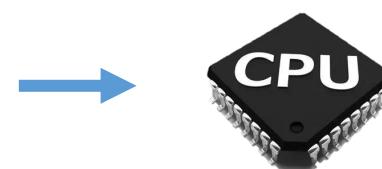


# 操作系统的运行机制

预备知识：程序是如何运行的？

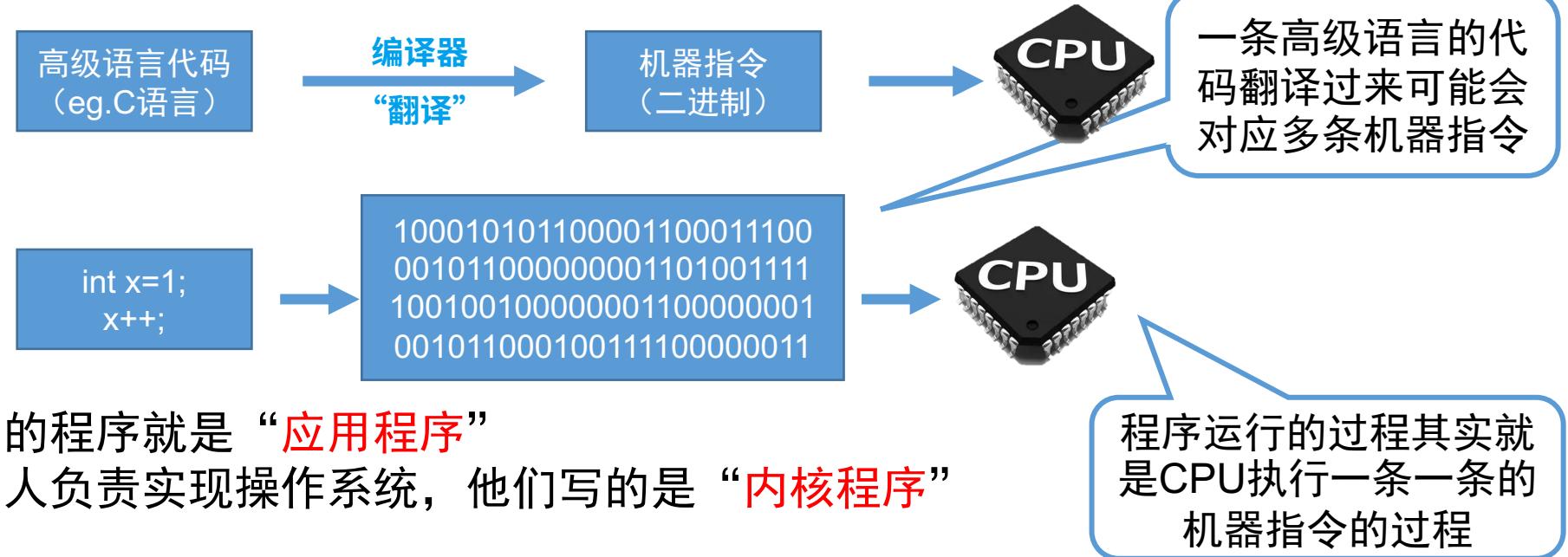


一条高级语言的代码翻译过来可能会对应多条机器指令



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

# 操作系统的运行机制——内核程序vs应用程序



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**”

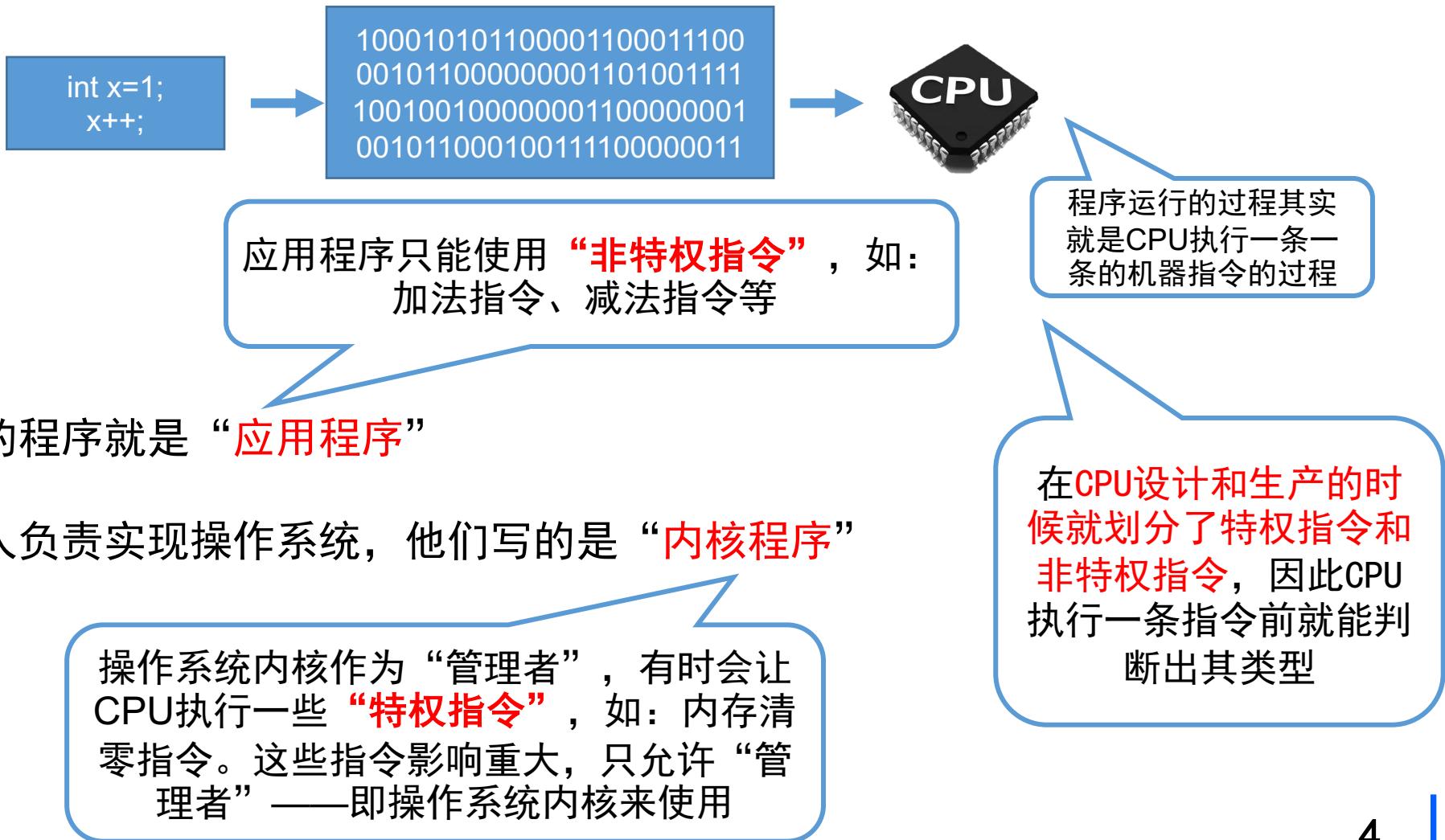
由很多内核程序组成了“**操作系统内核**”，或简称“**内核（Kernel）**”

**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**

甚至可以说，一个操作系统只要有内核就够了（eg: Docker—>仅需Linux内核）

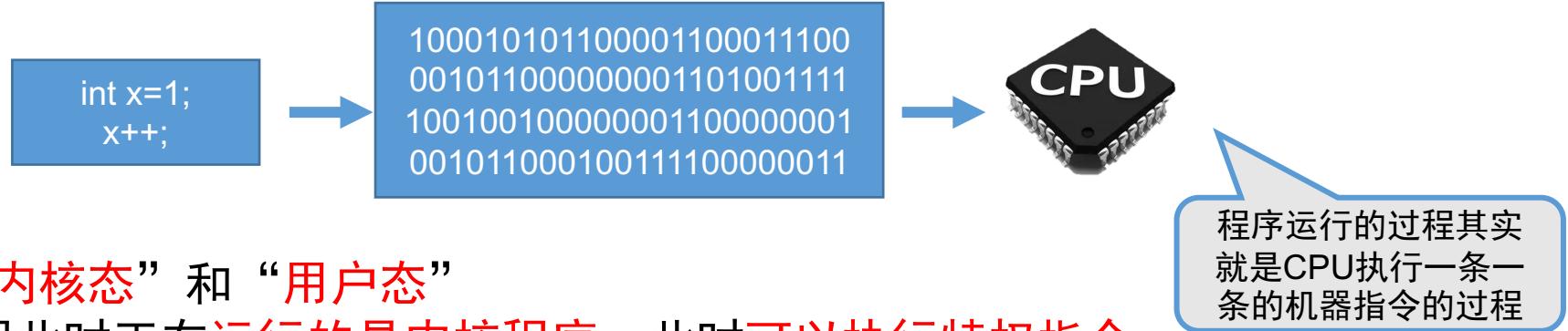
操作系统的功能未必都在内核中，如图形化用户界面GUI

# 操作系统的运行机制——特权指令vs非特权指令



# 操作系统的运行机制——内核态vs用户态

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？



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**

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**”

别名：内核态=核心态=管态；用户态=目态

# 操作系统的运行机制——内核态和用户态的切换

**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

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…
- ⑥CPU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

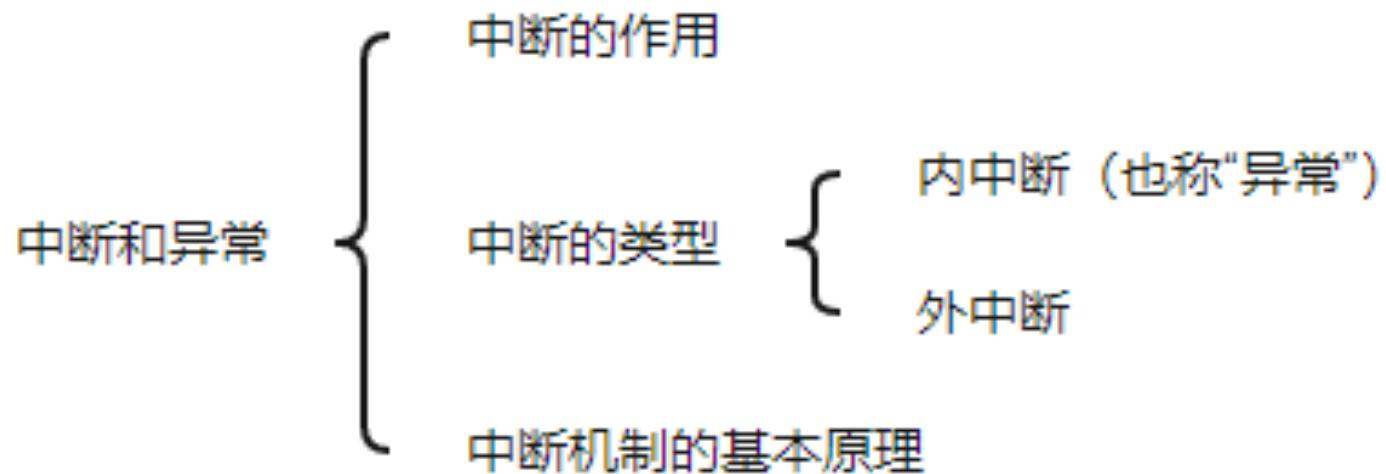
操作系统内核在让出CPU之前，会用一条**特权指令**把PSW的标志位设置为“**用户态**”

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 基本概念

中断和异常

# 中断和异常



## 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

## 中断和异常——中断的类型

内中断：

与当前执行的指令**有关**，  
中断信号来源于CPU**内部**

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

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

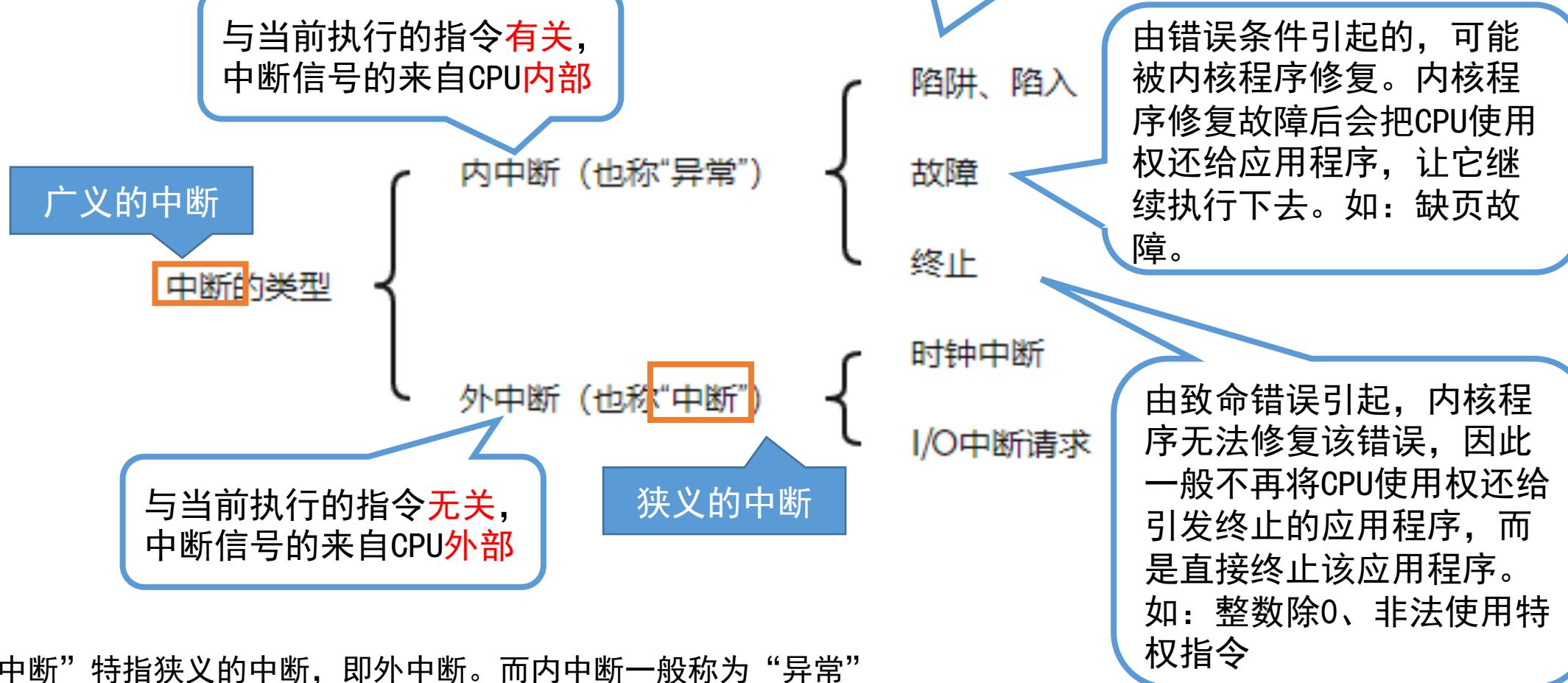
外中断：

与当前执行的指令**无关**，  
中断信号来源于CPU**外部**

例子1：时钟中断——由时钟部件发来的中断信号

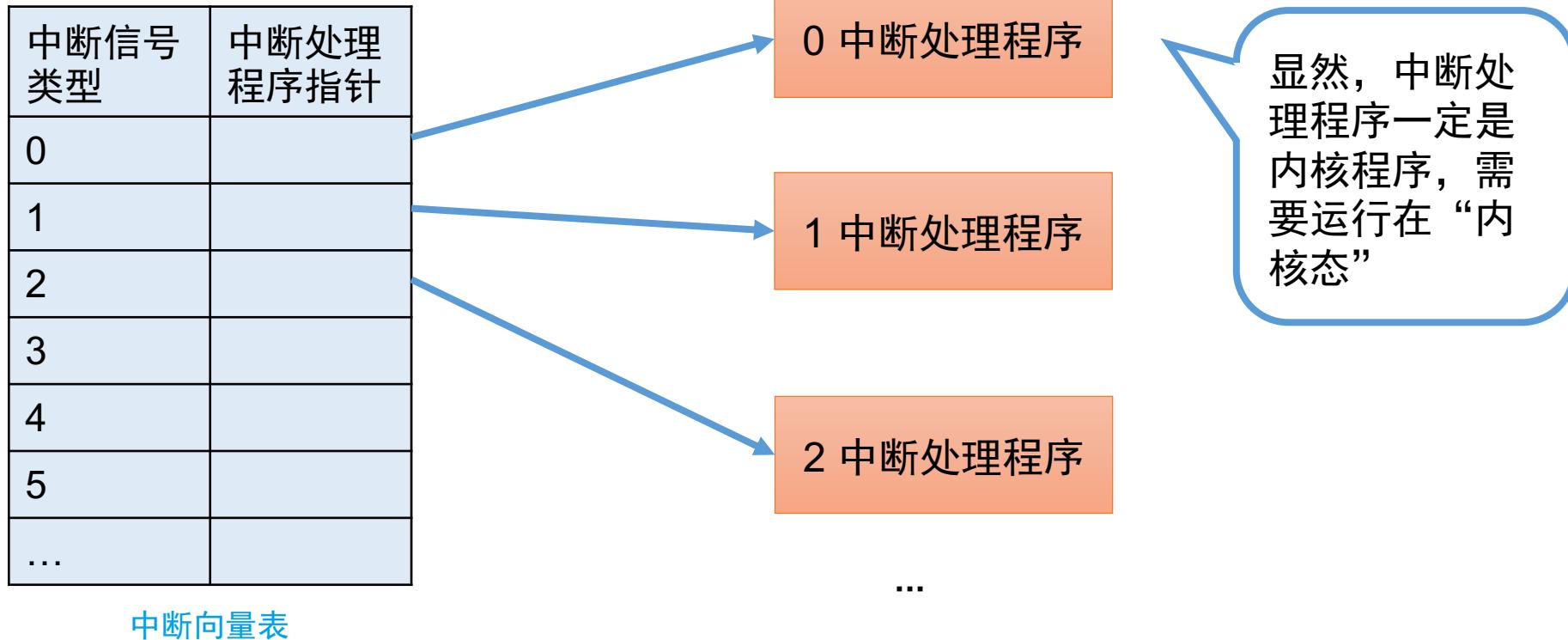
例子2：I/O中断——由输入/输出设备发来的中断信号

## 中断和异常——中断的类型



## 中断和异常——中断的基本原理

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。



# 操作系统的概念



系统调用



操作系统是计算机资源的管理者

是不是意味着应用程序**不具备**使用计算机资源的权限

如果有程序想使用硬件应该怎么做？

系统调用

安全措施

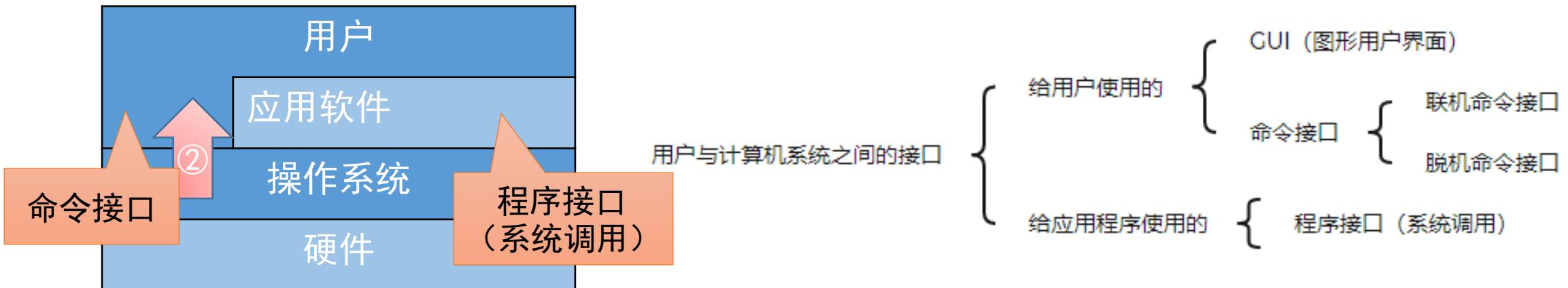
参数

应用程序

# 系统调用

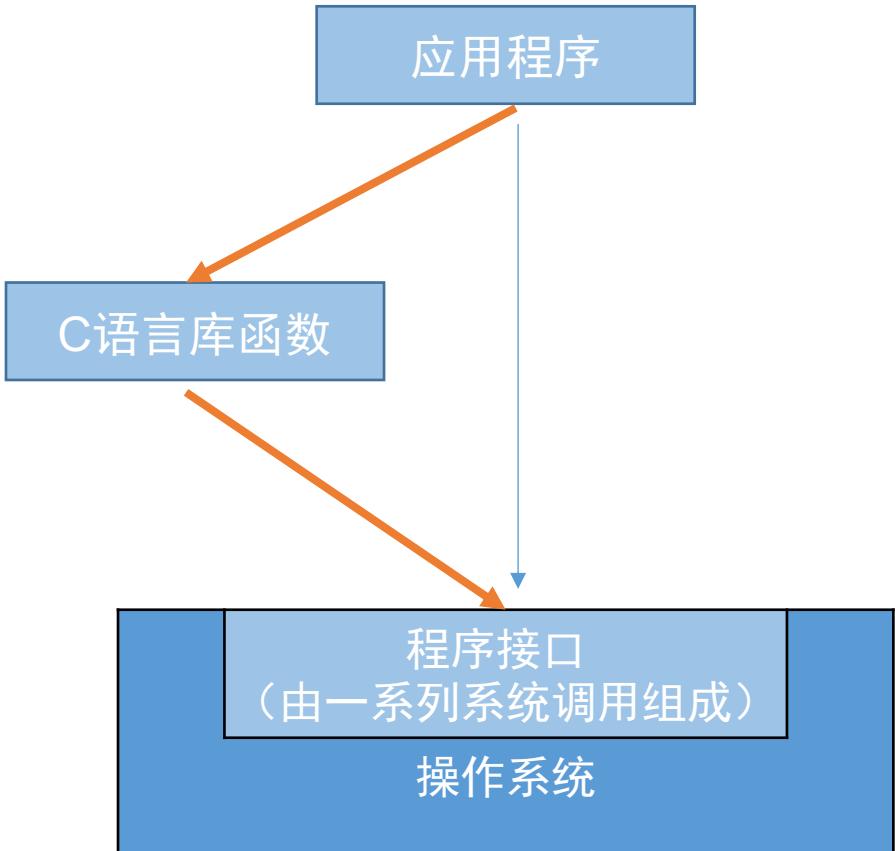
知识点回顾：

操作系统作为用户和计算机硬件之间的接口，需要向上提供一些简单易用的服务。主要包括命令接口和程序接口。其中，程序接口由一组**系统调用**组成。



“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**

# 系统调用——系统调用和库函数的区别

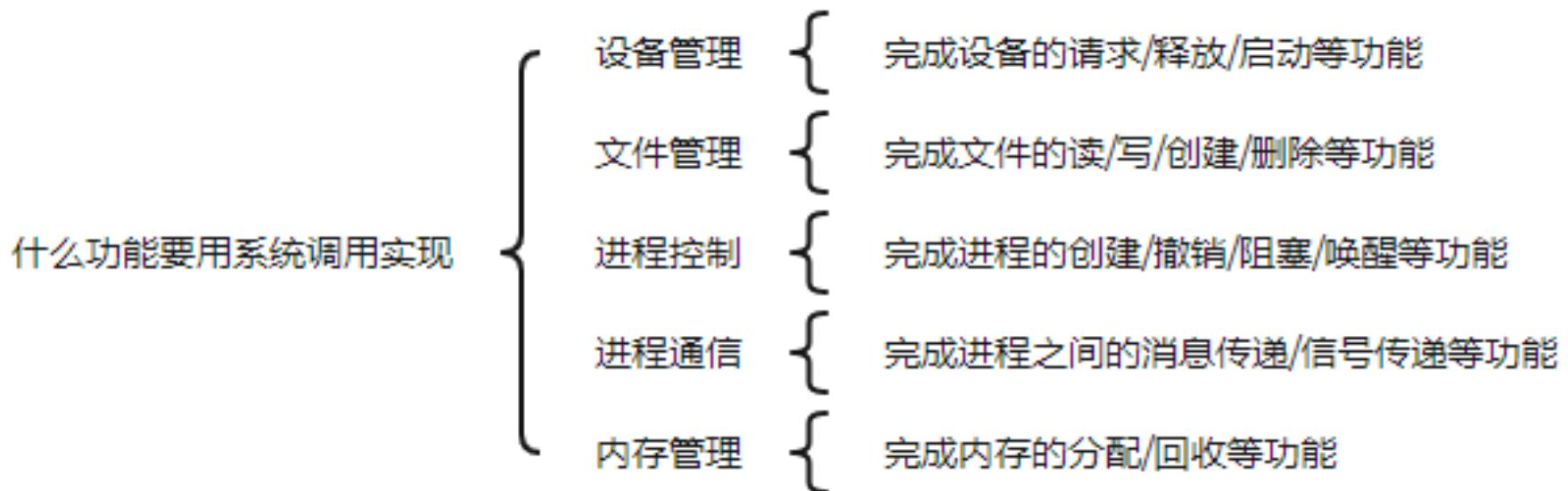


普通应用程序	可直接进行系统调用，也可使用库函数。有的库函数涉及系统调用，有的不涉及
编程语言	向上提供库函数。有时会将系统调用封装成库函数，以隐藏系统调用的一些细节，使程序员编程更加方便。
操作系统	向上提供系统调用，使得上层程序能请求内核的服务
裸机	

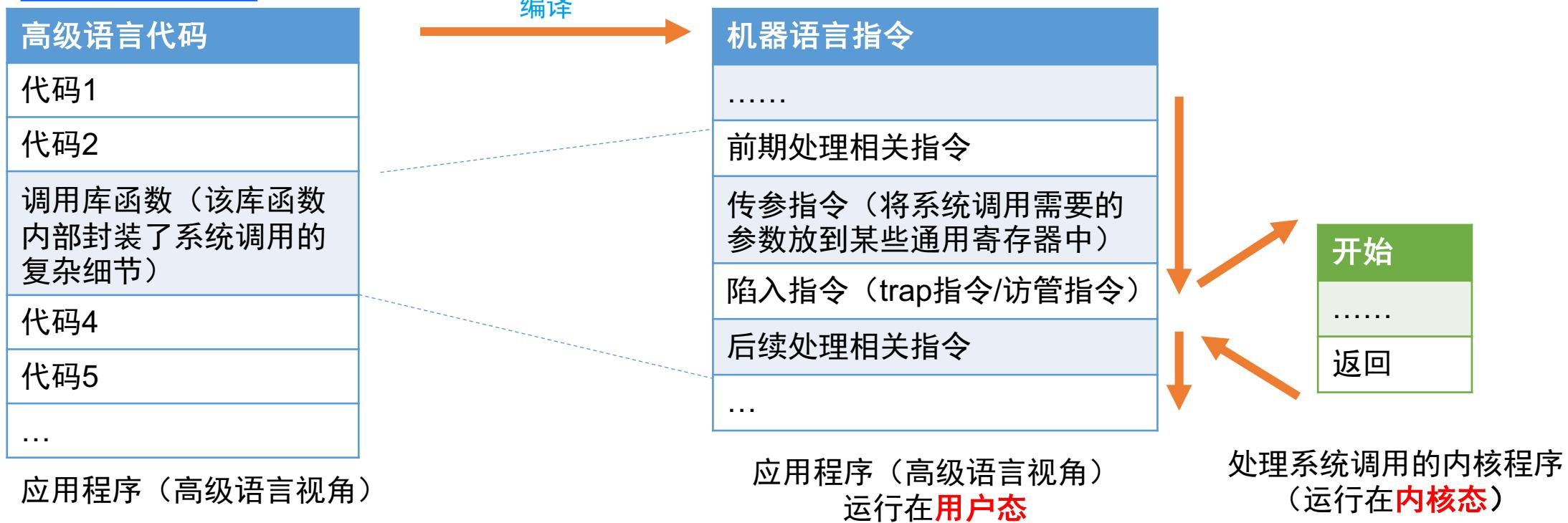
不涉及系统调用的库函数：如的“取绝对值”的函数涉及系统调用的库函数：如“创建一个新文件”的函数

## 系统调用——什么功能要用系统调用实现？

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。



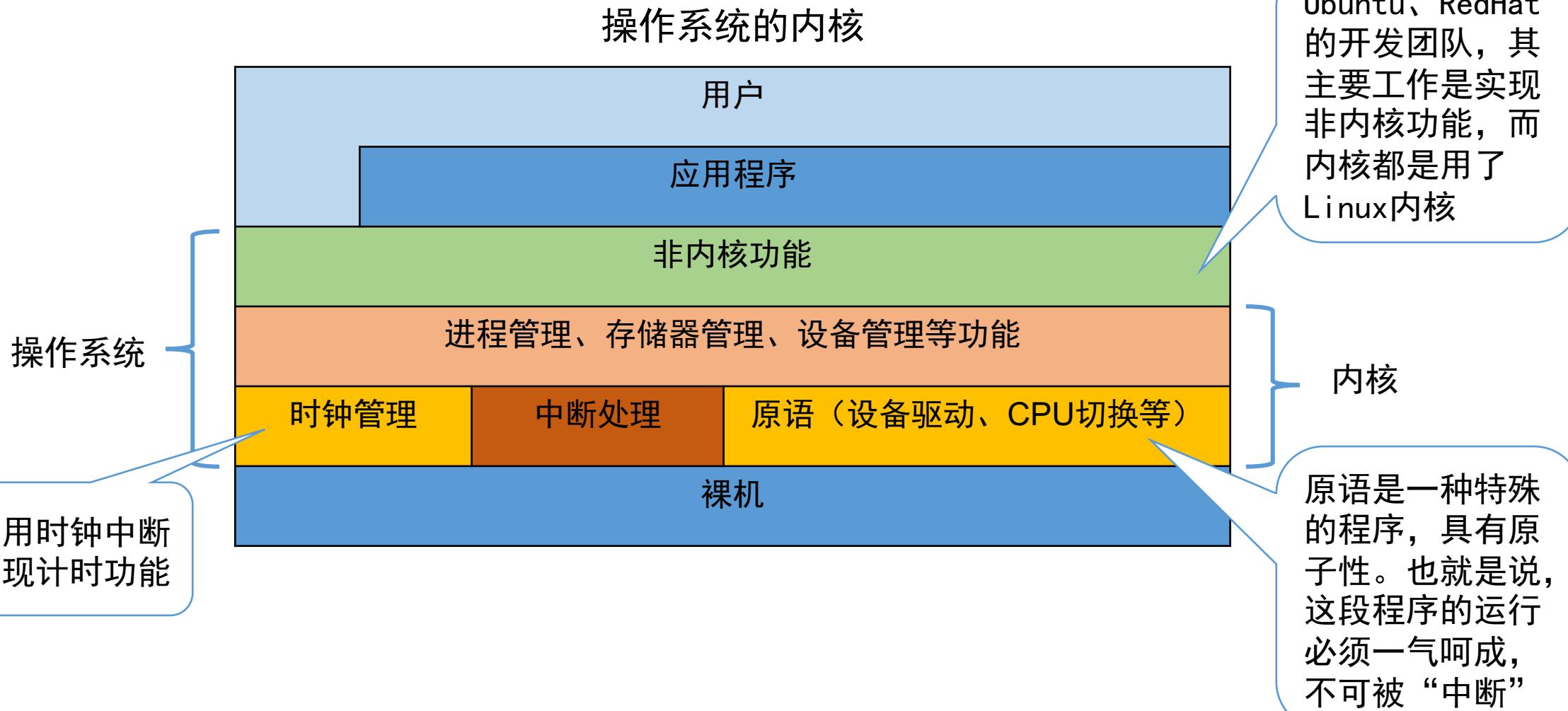
# 系统调用——什么功能要用系统调用实现？



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**）->返回应用程序

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

# 操作系统的体系结构



# 操作系统的体系结构

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。

操作系统内核

时钟管理 —— 实现计时功能

中断处理 —— 负责实现中断机制

是一种特殊的程序

原语 处于操作系统最底层，是最接近硬件的部分

这种程序的运行具有原子性 —— 其运行只能一气呵成，不可中断

运行时间较短、调用频繁

对系统资源进行管理的功能

进程管理

存储器管理

设备管理

与硬件关联较紧密的模块

这些管理工作更多的  
是对数据结构的操作，  
不会直接涉及硬件

拜拜