内容回顾

1. 操作系统OS定义：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。

2. 操作系统的发展：

•单道批处理：自动性、顺序性、单道性

•多道批处理：多道：内存中同时存放几个作业；

宏观：并行运行，都处于运行状态；

微观：串行运行，各作业交替使用CPU。

•分时：　　交互性：用户与系统进行人机对话。

多路性：多用户同时在各自终端上使用同一CPU。

独立性：用户可彼此独立操作，互不干扰，互不混淆。

及时性：用户在短时间内可得到系统的及时回答

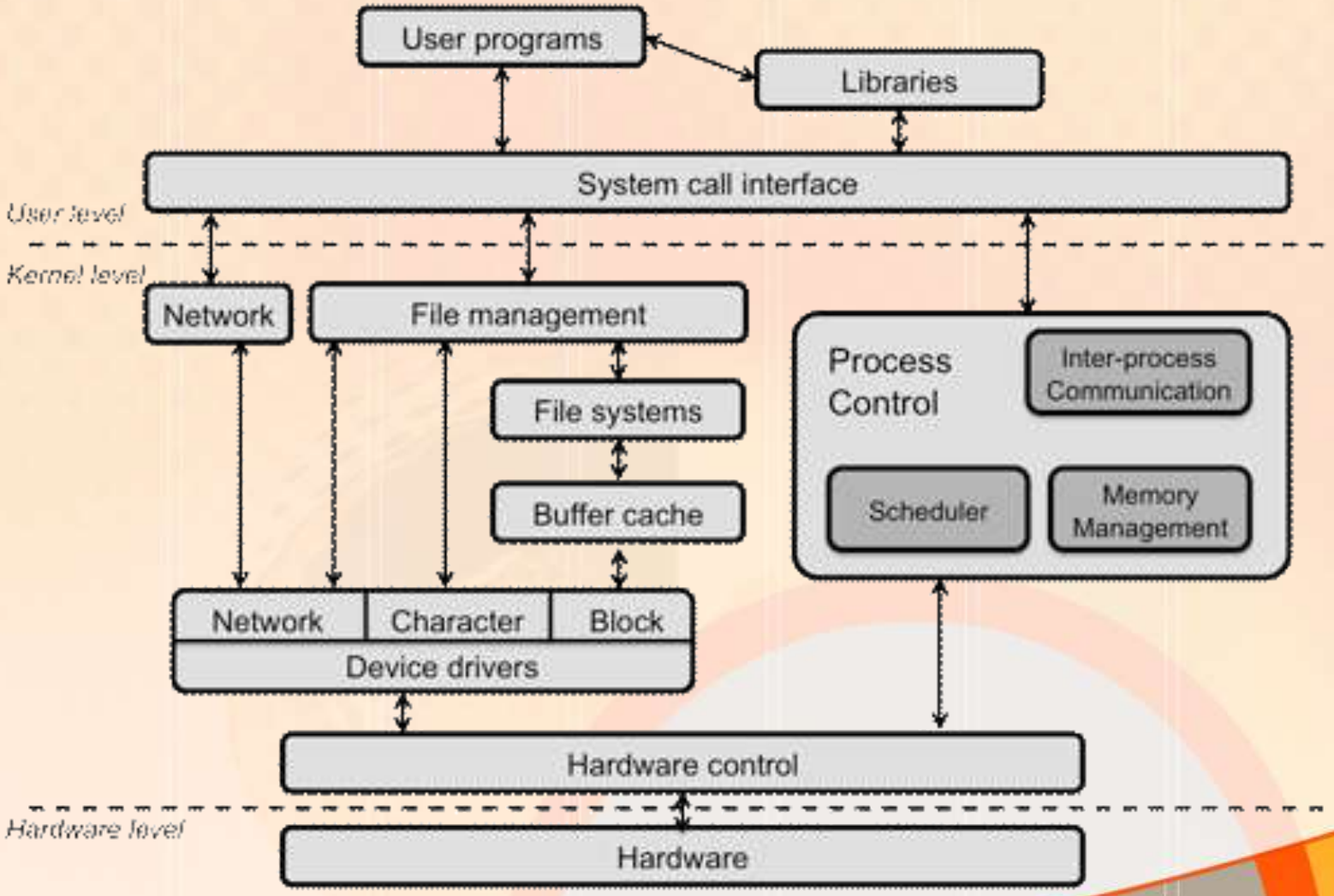
•实时：实时性、高可靠性

3. 操作系统的基本特性：并发、共享、虚拟、异步。

1.4 操作系统的功能

- **1.4.1 处理机管理**
- **1.4.2 存储管理**
- **1.4.3 设备管理**
- **1.4.4 文件管理**
- **1.4.5 用户接口**

操作系统的层次结构（OS的组成）



操作系统的层次结构（OS的组成）

- **操作系统的组成 ：**
  1. **管理模块：针对不同管理对象的程序模块（通常称为操作系统核心kernel）**
  2. **用户接口：如外壳(shell)、窗口系统**
     - **在shell中，通过运行其他程序来完成各种功能**

1.4.1 处理机管理

- **完成处理机资源的分配调度等功能。**
  1. **进程控制：创建、撤销、挂起、改变运行优先级等（为作业创建进程、撤销已结束的进程，以及控制进程在运行过程中的状态转换）；**
  2. **进程同步：协调并发多个进程之间的推进步骤，以协调资源共享（进程互斥方式、进程同步方式）；**
  3. **进程通信：进程之间传送数据，以协调进程间的协作（例如：输入进程、计算进程和打印进程这三个相互合作的进程之间会进行信息交换）；**
  4. **调度：作业或进程的运行切换。作业调度是从后备队列中选择出若干个作业，为其分配资源（内存）建立进程；进程调度是从就绪队列中选出一个新进程投入执行。**

1.4.2 存储管理

- **管理目标：提高利用率、方便用户使用、提供足够的存储空间、方便进程并发运行。**
  1. **存储分配与回收**
  2. **存储保护：保证进程间互不干扰、相互保密；**
  3. **地址映射（变换）：进程逻辑地址到内存物理地址的映射；**
  4. **内存扩充（覆盖、交换和虚拟存储）：提高内存利用率、扩大进程的内存空间。**

1.4.3 设备管理

- **目标：方便设备使用、提高CPU与I/O设备利用率**
  - **设备操作：利用设备驱动程序（通常在内核中）完成对设备的操作。**
  - **设备独立性(device independence)：提供统一的I/O设备接口，使应用程序独立于物理设备，在同样的接口和操作下完成不同的内容（如FAX Modem作为Windows上的打印机设备）。**
  - **设备分配与回收：在多用户间共享I/O设备资源。**

1.4.4 文件管理

- **解决软件资源的存储、共享、保密和保护。**

  1.文件存储空间管理：

  由文件系统对诸多文件及文件的存储空间，实施统一的管理。其主要任务是为每个文件分配必要的外存空间，提高外存的利用率，并能有助于提高文件系统的运行速度。

  2. 目录管理：

  解决文件检索问题。为了使用户能够方便地在外存上找到自己需要的文件，通常由系统为每个文件建立一个目录项，包括：文件名、文件属性、文件在磁盘上的物理位置等。

  3.文件的读写管理和存取控制：

  根据用户需求，从外存中读取数据，或将数据写入外存。

  文件保护：防止未经核准的用户存取文件、冒名顶替存取文件、以不正确的方式使用文件。

  4. 软件管理：软件的版本、相互依赖关系、安装和卸载等。

1.4.5 用户接口

- **目标：提供一个友好的用户访问操作系统的接口。通常以命令或系统调用的形式提供给用户：**
  - **命令接口：为联机用户提供的，由一组键盘操作命令及命令解释程序所组成.又分联机和脱机用户接口.**
    (1)联机用户接口.这是为联机用户提供的,它由一组键盘操作命令及命令解释程序所组成。
    (2)脱机用户接口。 该接口是为批处理作业的用户提供的，故也称为批处理用户接口.该接口由一组作业控制语言JCL组成。
  - **程序接口：为用户程序在执行中访问系统资源而设置的，是用户程序取得操作系统服务的路径。它由一组系统调用组成。**
  - **图形接口：用户可用鼠标或通过菜单和对话框，来完成对应用程序和文件的操作。**

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

一、无结构操作系统

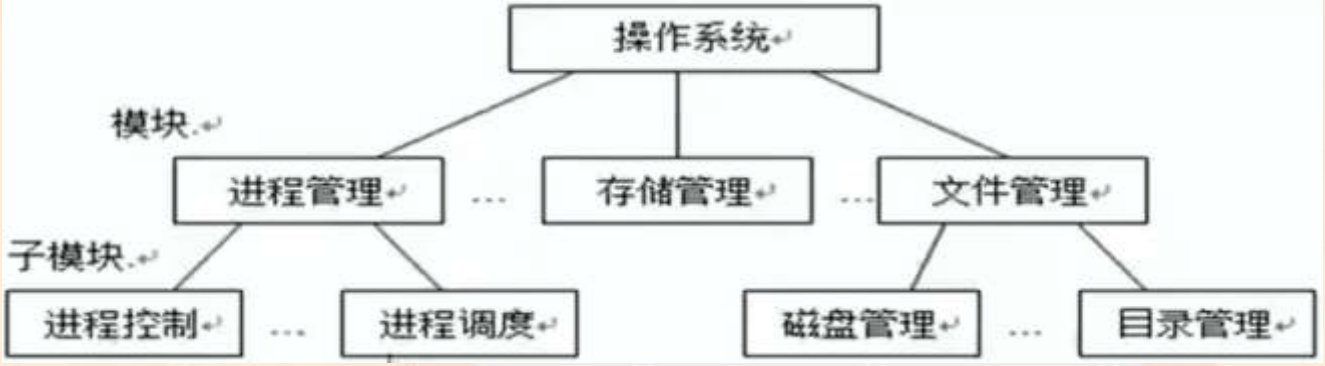
OS是由众多的过程直接构成，各过程之间可相互调用，但OS内部不存在任何结构，所以这种OS是无结构的，又称为整体系统结构。

缺点：

即庞大又杂乱，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。

二、模块化操作系统结构

OS采用“模块化程序设计”技术，按其功能划分为若干个独立地模块，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，对较大模块又可按照子功能进一步细分下去。



优点：

1. 提高了OS设计的正确性、可理解性
2. 容易扩充和维护
3. 加速了OS的开发过程

缺点：

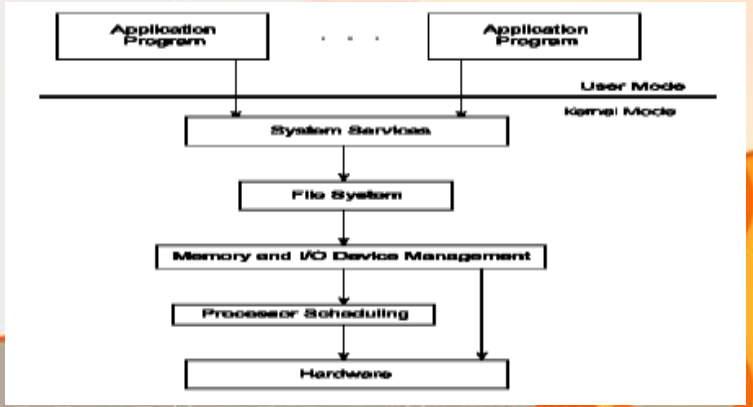
1. 模块及接口划分较为困难
2. 从功能上划分模块，没有区别对待共享资源和独占资源
3. 由于管理的差异，使得OS结构变得不够清晰

三、分层式操作系统结构

分层式OS结构对模块化结构进行了改进，它按照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结构成为分层式OS结构。

特点：

1. 每一步设计都建立在可靠的基础上，结构更清晰
2. 调试和验证更容易，正确性更高
3. 缺点：系统效率更低了



四、微内核的OS结构

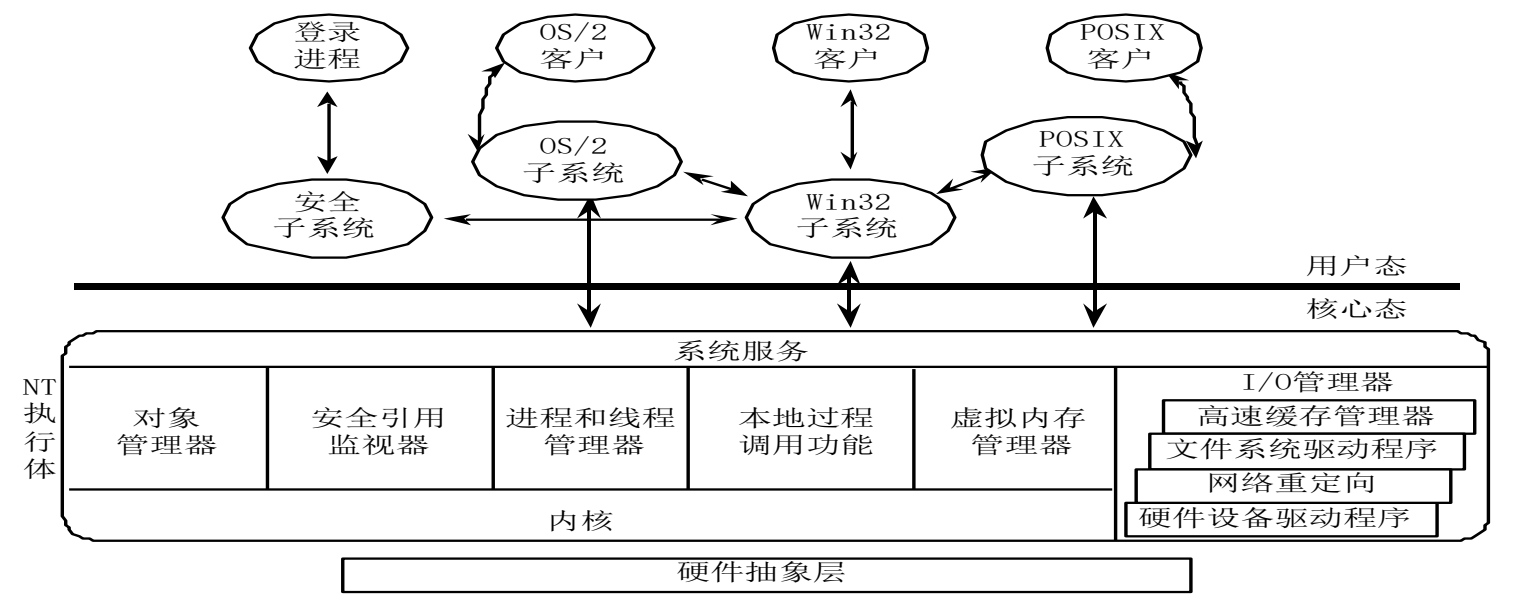
在OS内核中只留下一些最基本的功能，而将其他服务分离出去，由工作在用户态下的进程来实现，形成所谓“客户/服务器”模式。客户进程可通过内核向服务器进程发送请求，以获取OS的服务。

1. 足够小的内核
2. 基于客户/服务器模式
3. 应用“机制与策略分离”原理
4. 采用面向对象技术

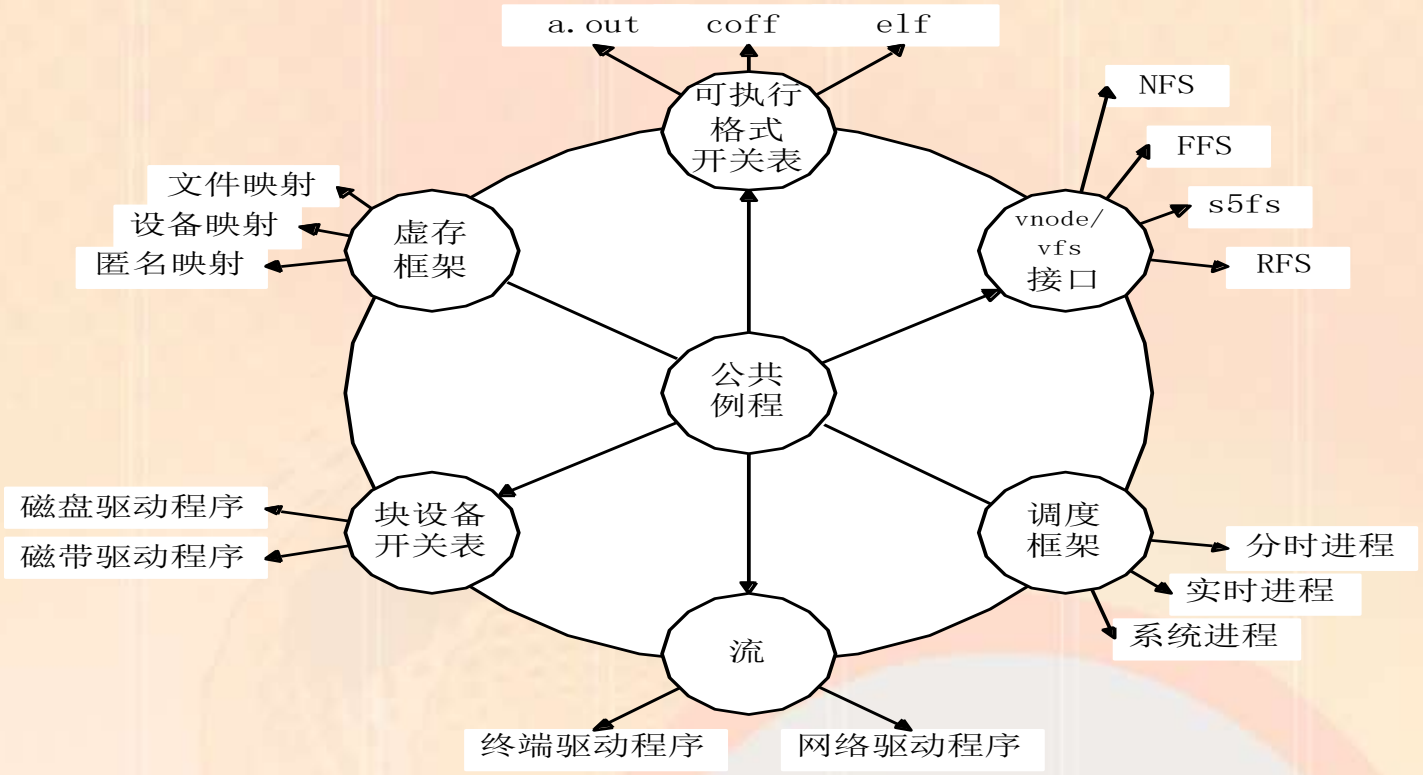
特点

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



Windows NT体系结构


a.out
coff
elf
可执行格式开关表
NFS
FFS
s5fs
RFS
vnode/vfs接口
文件映射
设备映射
匿名映射
虚存框架
公共例程
磁盘驱动程序
磁带驱动程序
块设备开关表
调度框架
分时进程
实时进程
系统进程
流
终端驱动程序
网络驱动程序

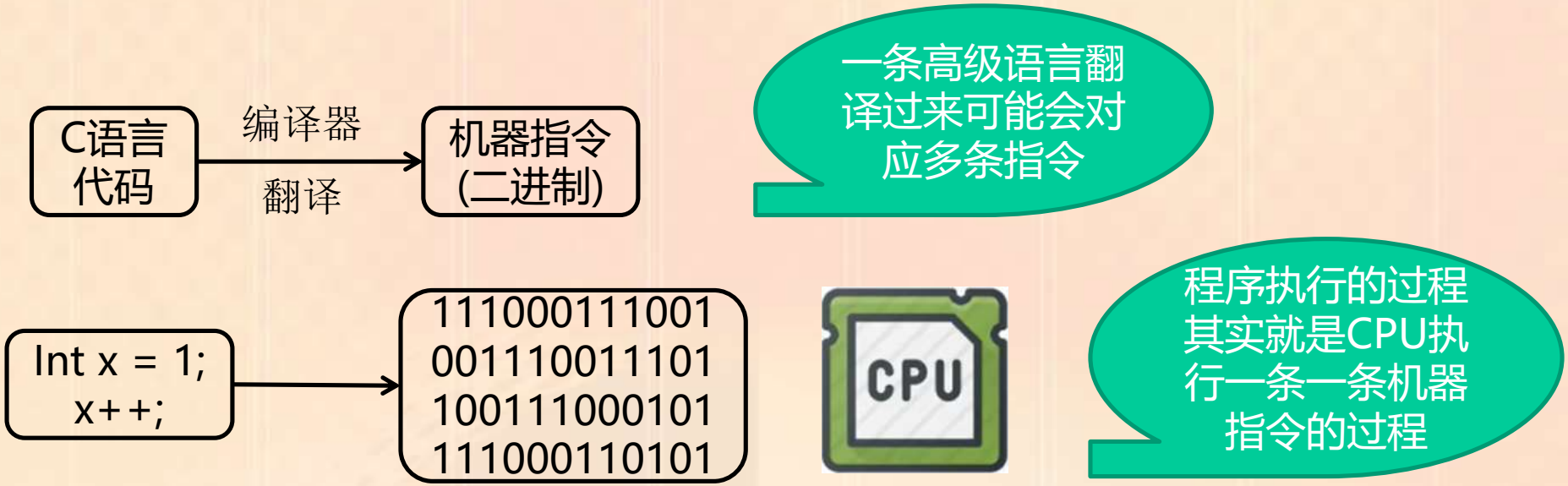
现代UNIX结构

常见的OS

- MS DOS
- MS Windows
- UNIX
- Linux
- 手持系统（handheld system）：安卓、苹果系统
- 嵌入式操作系统 (Embedded OS）：汽车导航系统

1.6 双模式操作

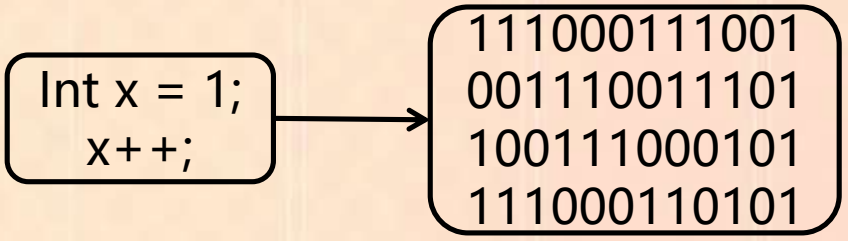
内核模式VS应用程序



普通程序员写的程序其实就是”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”，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（e.g: 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

1.6 双模式操作

特权指令VS非特权指令



程序执行的过程其实就是CPU执行一条一条机器指令的过程

应用程序只能使用非特权指令，如：加法指令、减法指令等

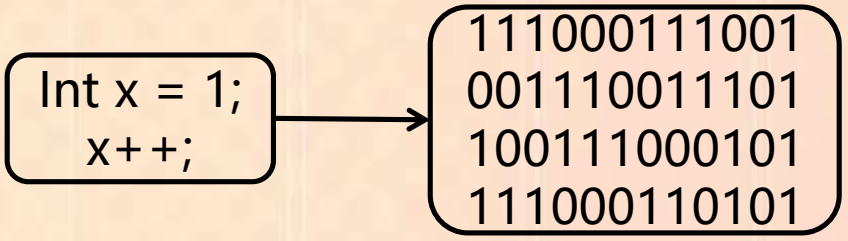
普通程序员写的程序其实就是"应用程序"

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"

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。

1.6 双模式操作

内核态VS用户态



程序执行的过程其实就是CPU执行一条一条机器指令的过程

CPU能够判断出指令类型，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”

CPU处于**内核态**的时候，说明CPU此时正在运行**内核程序**，此时可以执行**特权指令**。

CPU处于**用户态**的时候，说明CPU此时正在运行**应用程序**，此时可以执行**非特权指令**。

1.6 双模式操作

- 硬件支持至少两种运行模式:
  1. 用户模式（又叫用户态、目态）– 执行普通用户的应用程序。
  2. 监控模式（Monitor mode，也称内核态、系统态、监控态、管态、特权模式) – 执行操作系统核心代码

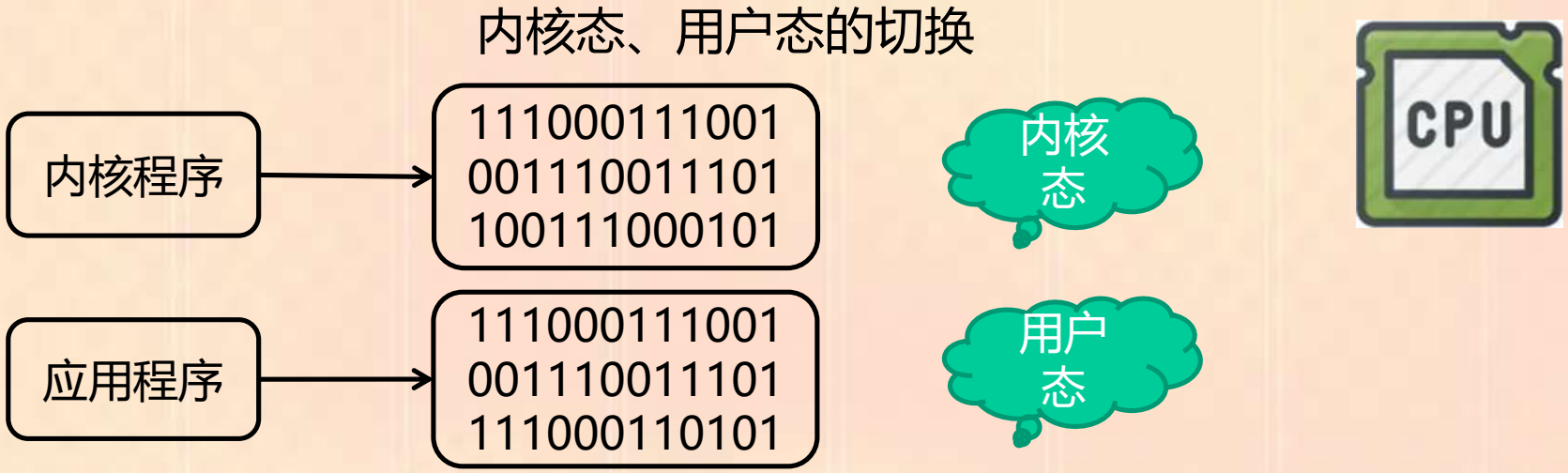
内核态=核心态=管态：执行特权指令

用户态=目态：执行非特权指令

如何区分双模式及其切换

- CPU中有一个寄存器叫做程序状态字寄存器（PSW），其中有个二进制位，内核态 (0) 或用户态 (1)

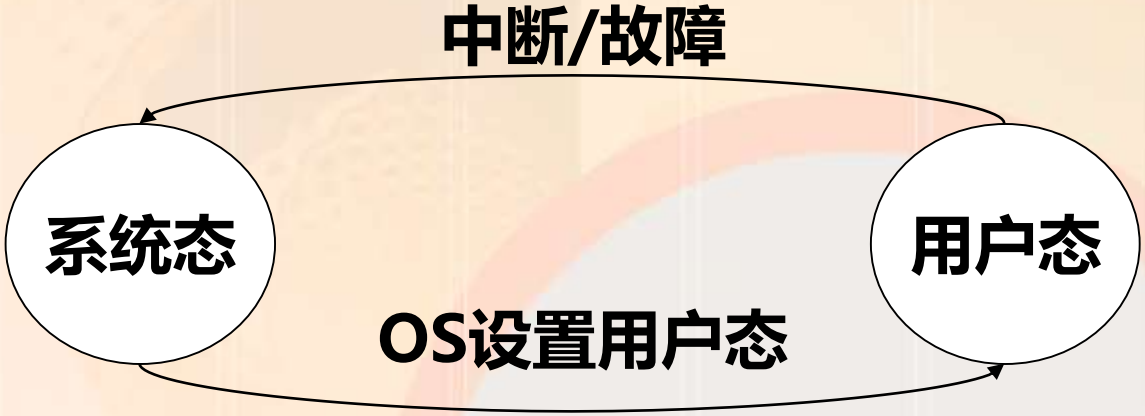
1.6 双模式操作



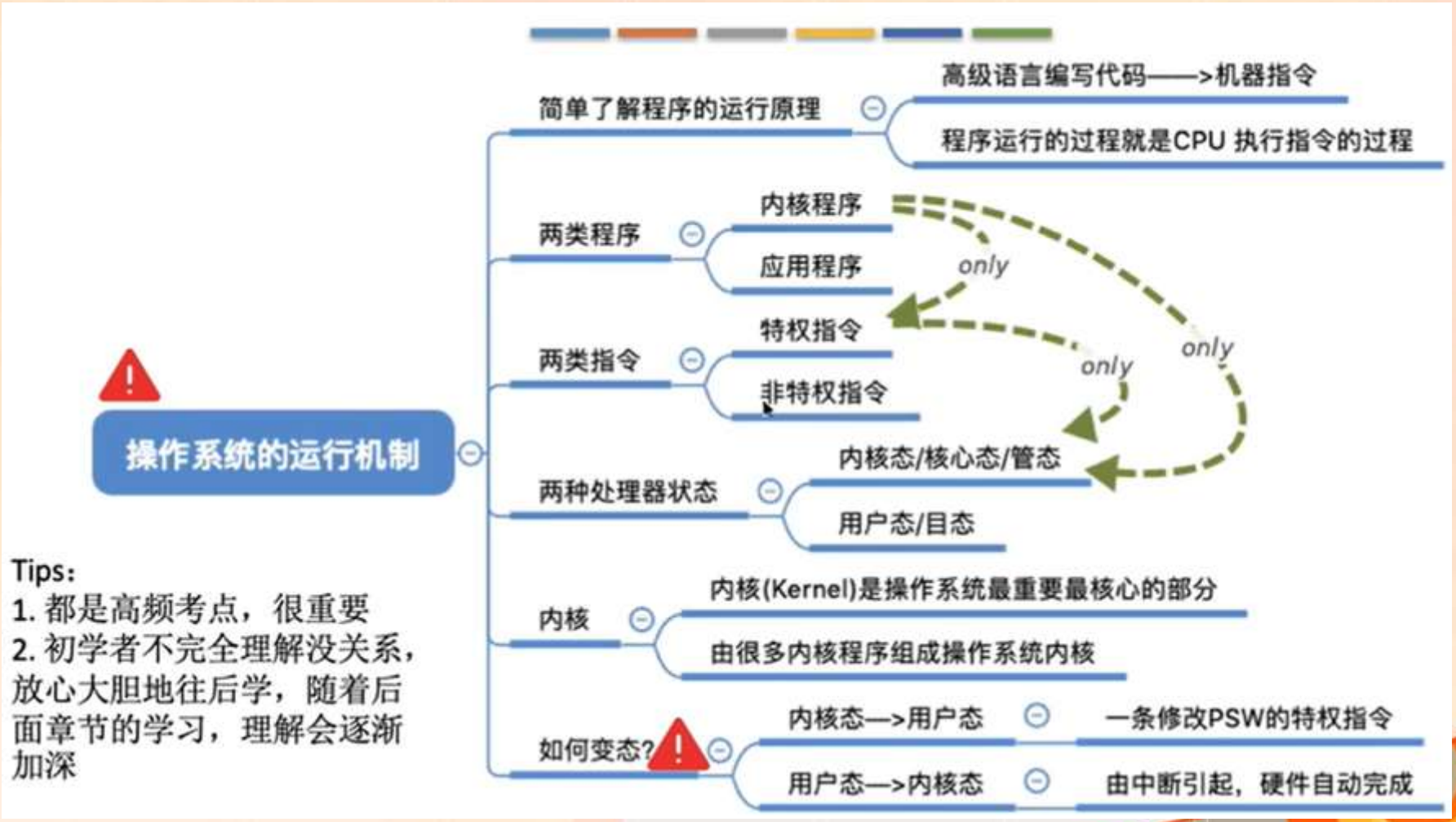
1. 刚开机时，CPU为“内核态”，操作系统内核程序先上CPU运行。
2. 开机完成后，用户可以启动某个应用程序。
3. 操作系统内核程序在合适的时候主动让出CPU，让该应用程序上CPU运行。（此时，操作系统内核在让出CPU之前，会用一条特权指令，把PSW的标志位设置为“用户态”）
4. 应用程序运行在“用户态”
5. 一位猥琐的黑客在应用程序中植入了一条特权指令。企图破坏系统。
6. CPU发现接下来要执行的这条指令是特权指令，但是自己处于“用户态”
7. 这个非法事件会引发一个中断信号
8. 中断使得操作系统内核再次拿回CPU的控制权。
9. 操作系统会对引发中断的事件进行处理，处理完了再把CPU的使用权交给别的应用程序。

- 内核态--用户态: 执行一条特权指令--修改PSW特权指令为用户态，这个动作意味着操作系统主动让出CPU的使用权。
- 用户态--内核态: 由中断引发，由CPU硬件自动完成变化状态的过程，触发中断信号意味着操作系统将强行夺回CPU的使用权，并停止运行当前的应用程序，转而运行内核程序。

  (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。共性：但凡需要操作系统介入的地方，都会触发中断信号)
- 当用户程序需要操作系统的服务（通过系统调用），它必须由用户态切换到核心态。



- 双重模式操作可以确保系统和用户程序不受错误的用户程序的影响。
- 实现方法：将能引起损害的机器指令作为特权指令。特权指令只能在系统态下运行。如果用户模式下试图执行特权指令，那么硬件认为该指令非法，并以陷阱的形式通知操作系统。
- 由管态转换到目态是特权指令，其他的特权指令有I/O控制、定时器管理和中断管理。
- **常见的特权指令有以下几种:**
  （1）有关对I/O设备使用的指令 如启动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。
  （2）有关访问程序状态的指令 如对程序状态字（PSW）的指令等。
  （3）存取特殊寄存器指令 如存取中断寄存器、时钟寄存器等指令。
  （4）其他指令



Tips:
1. 都是高频考点，很重要
2. 初学者不完全理解没关系，放心大胆地往后学，随着后面章节的学习，理解会逐渐加深

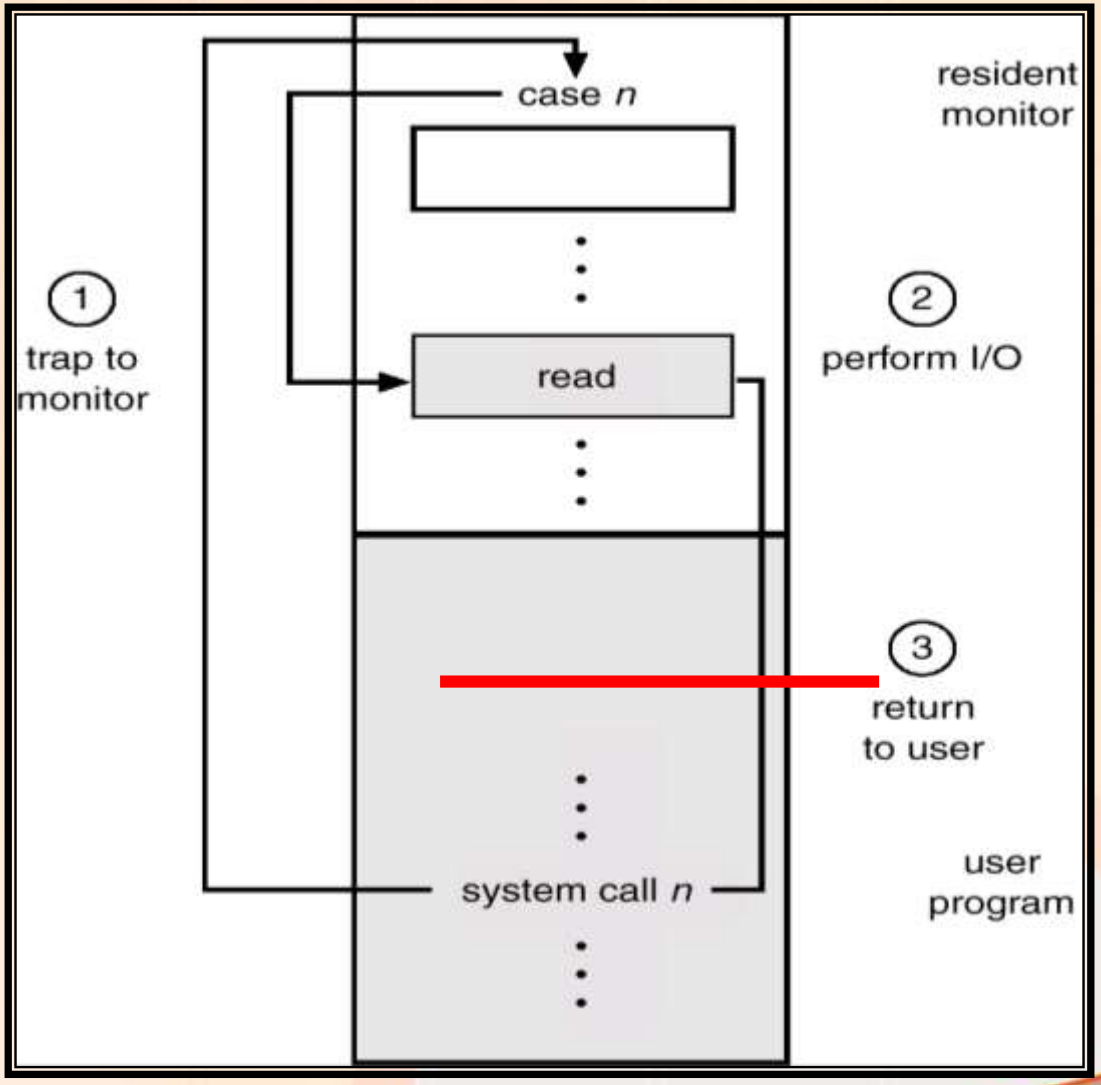
- *例：下列选项中在用户态下执行的有：*
- *A 命令解释程序*
- *B 缺页处理程序*
- *C 进程调度程序*
- *D 时钟中断处理程序*

- *例：下列选项中，不可能在用户态发生的事情是*
- *A 系统调用　B 外部中断*
- *C 进程切换　D 缺页*

1.6.2 I/O 保护

- **所有的 I/O 指令是特权指令（ privileged instructions ）。**
- **必须确保所有的用户程序不能获得管态下操作系统所拥有的对系统的控制权。**

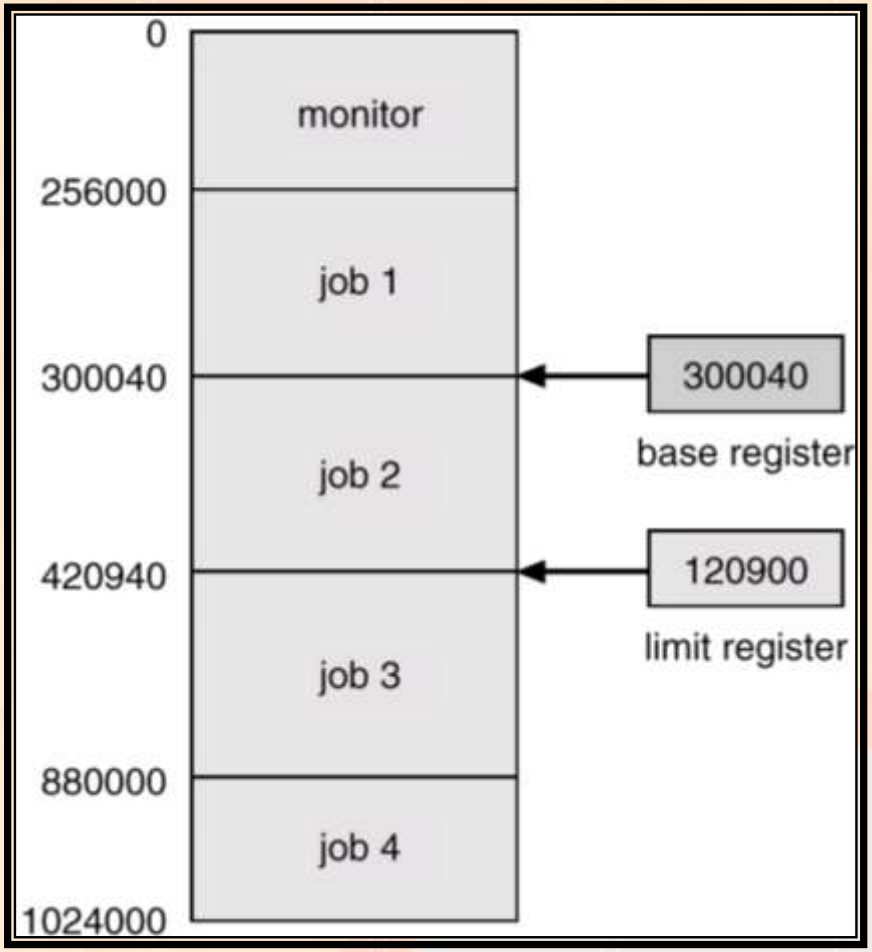
用户通过系统调用执行 I/O



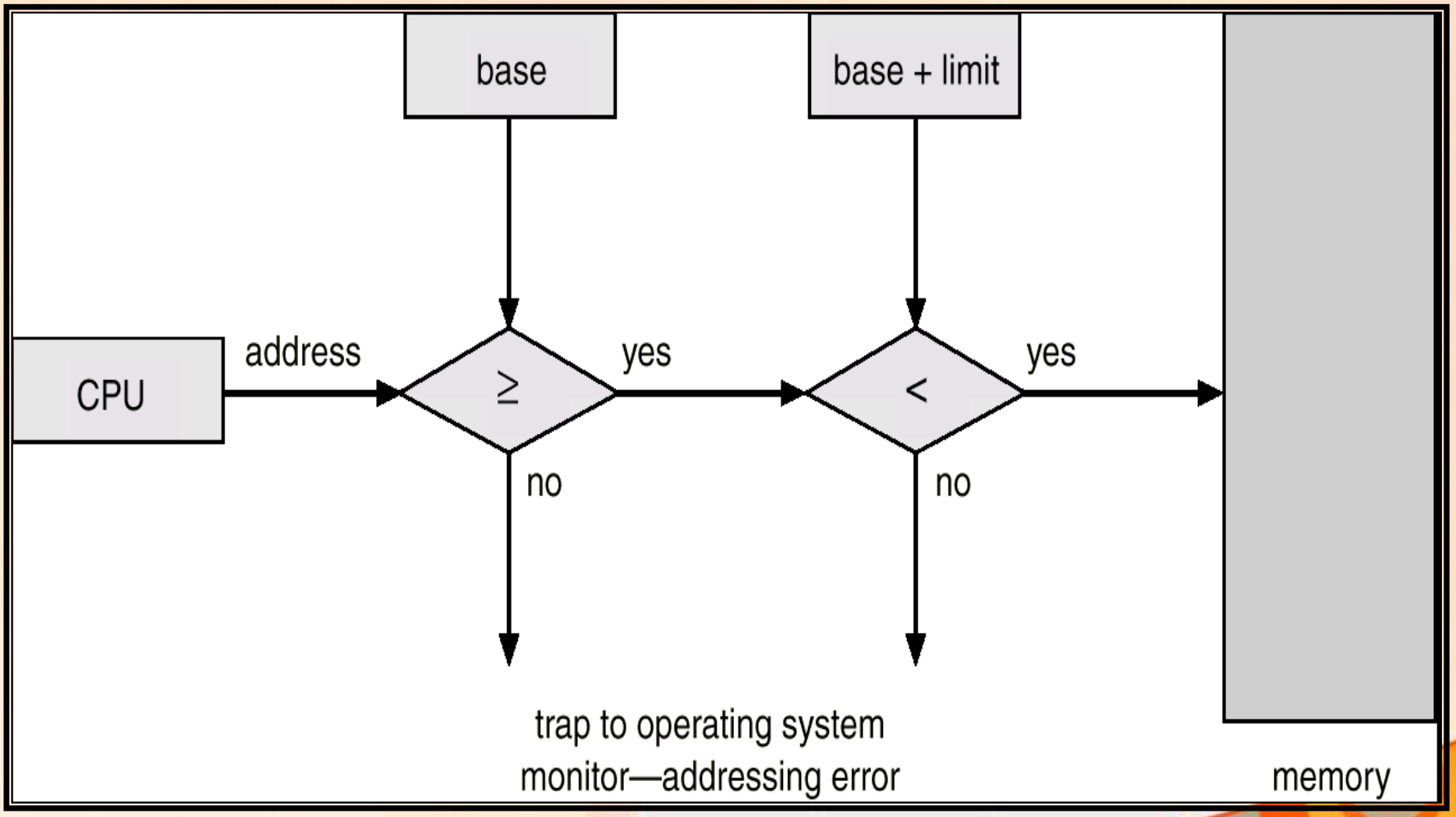
1.6.3 存储保护

- 存储保护是指：
  - 某进程不可以访问其它进程的地址空间（非共享情况）
  - 用户进程必须通过系统调用访问OS核心区。
- 为了实现存储保护，在硬件上增加两个寄存器：
  - 基地址寄存器 – 存放本进程最低的内存物理地址
  - 界限寄存器 – 存放本进程的存储区域大小

基地址和界限寄存器的使用



硬件对地址的保护


base
base + limit
CPU
address
≥
yes
<
yes
no
no
trap to operating system
monitor—addressing error
memory

硬件保护原则

- **当操作系统在核心态下执行时，操作系统能够无限制地访问核心和所有用户进程的存储空间。**

- **修改基地址和界限地址寄存器内容的指令属特权指令。**

1.6.4 CPU 保护

- 用户进程被派发CPU后，若程序是死循环，始终不执行系统调用，则OS无法获得CPU的控制权，系统必须避免这种情况发生。有如下机制：
- 硬件设置时钟 – 定时中断，该中断使操作系统获得CPU控制权。
  - 定时是通过对每个时钟脉冲计数获得的，每来一个脉冲，预定的计数值减1，
  - 当计数值减到0，发生定时器中断

- 定时时钟是实现分时（time-sharing)操作系统的硬件基础。
  - 设定每N毫秒（时间片）时钟中断一次，即每个进程可使用处理器N毫秒，然后无论完成与否都出让CPU，排队等待下个时间片。
- 修改定时时钟的指令为特权指令。