嵌入式系统
An Introduction to Embedded System

第五课 嵌入式实时操作系统（RTOS）简介

教师：蔡铭 cm@zju.edu.cn

浙江大学计算机学院人工智能研究所
航天科技—浙江大学基础软件研发中心

课程大纲

- 嵌入式实时操作系统概况
- 嵌入式实时操作系统特点
- 嵌入式实时操作系统功能简介

嵌入式实时系统

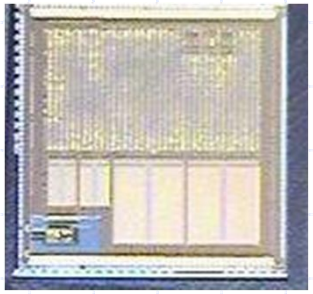
- 嵌入式系统往往对实时性提出较高的要求。
- 实时系统：指系统能够在限定的响应时间内提供所需水平的服务。（POSIX 1003.b ）
- 嵌入式实时系统可分为：
  - 强实时型：响应时间μs～ms级，如数控机床、医疗仪器；
  - 一般实时：响应时间ms～s级，如打印机、电子菜谱；
  - 弱实时型：响应时间s级以上，如工程机械控制。

背景分析

- 早期嵌入式系统：硬件所限
  - 汇编语言
  - 基本不采用操作系统
- 基础条件成熟
  - 硬件的提升

    ——微处理器性能提高、存储器容量增加
  - 软件技术快速发展

    ——编译器、操作系统、集成开发环境

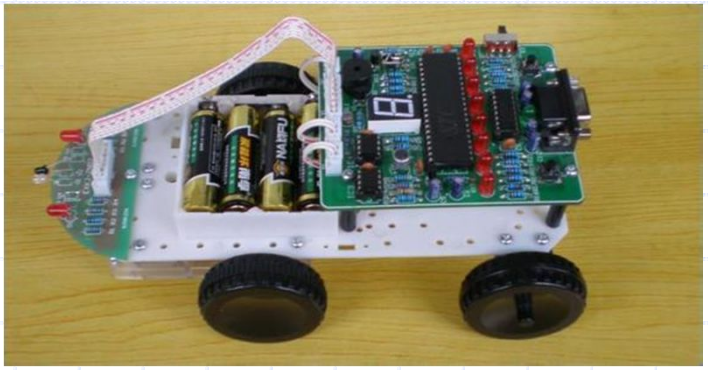


嵌入式操作系统概述－发展阶段（1/4）

- 嵌入式操作系统的发展主要经历了以下四个阶段：
- 无操作系统的阶段
  - 单芯片为核心
  - 具有与一些监测、伺服、指示设备相配合的功能
  - 一般没有明显的操作系统支持
  - 通过汇编语言编程对系统进行直接控制。
  - 主要特点
    - 系统结构和功能都相对单一，针对性强
    - 无操作系统支持
    - 几乎没有用户接口

嵌入式操作系统概述－发展阶段（2/4）

- 简单监控式的实时操作系统阶段
  - 以嵌入式处理器为基础
  - 以简单监控式的操作系统为核心
  - 主要特点：
    - 处理器种类繁多，通用性比较弱；
    - 开销小，效率高；
    - 一般配备系统仿真器，具有一定的兼容性和扩展性；
    - 用户界面不够友好，主要用来控制系统负载，以及监控应用程序运行。



■八十年代初：出现了以**VRTX**(1981)、**pSOS**等为代表的第一代系统（实时内核），提供了实时操作系统基本功能。

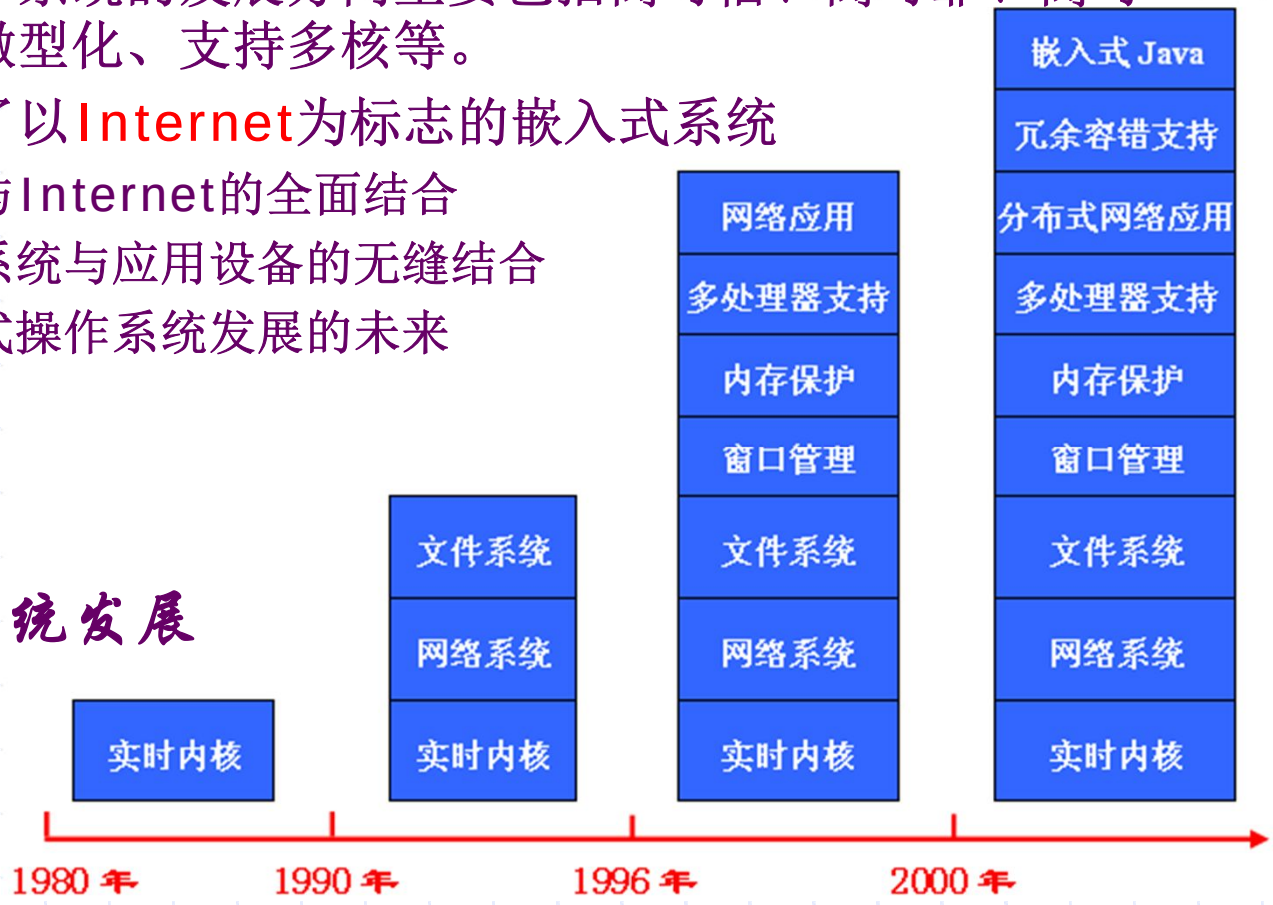
嵌入式操作系统概述－发展阶段（3/4）

- 通用的嵌入式实时操作系统阶段
  - 以通用型嵌入式操作系统为标志的嵌入式系统
  - 主要特点：
    - 运行在不同的微处理器
    - 具有强大的通用型操作系统的功能
      - 文件和目录管理
      - 多任务
      - 设备驱动支持
      - 网络支持
      - 图形窗口
      - 用户界面
    - 具有丰富的API和嵌入式应用软件

■八十年代后期到九十年代初期，出现以VxWorks、RTEMS、Nucleus PLUS、QNX、OSE为代表的第二代系统。

嵌入式操作系统概述－发展阶段（4/4）

- 二十世纪末，出现了以Integrity为代表的第三代系统，进一步在实时性、高可靠性、高可用性等方面提供了强有力的支持。
- 新一代实时操作系统的发展方向主要包括高可信、高可靠、高可用、高安全、微型化、支持多核等。
- 近年来，出现了以Internet为标志的嵌入式系统
  - 嵌入式系统与Internet的全面结合
  - 嵌入式操作系统与应用设备的无缝结合
  - 代表着嵌入式操作系统发展的未来



嵌入式实时操作系统发展

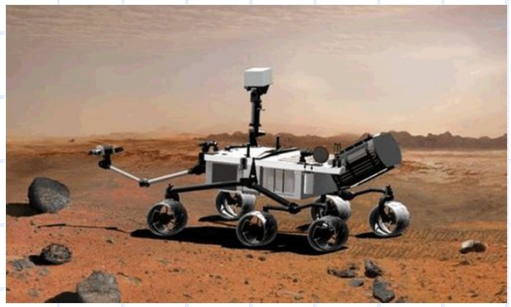
典型的嵌入式实时操作系统

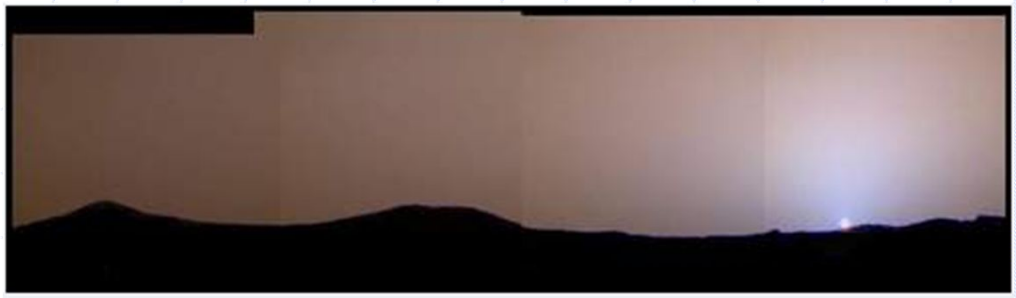
- 嵌入式实时操作系统数量众多，如：
  - VxWorks
  - Windows CE
  - pSOS
  - QNX
  - PalmOS
  - Nucleus
  - Android
  - RT-Linux
  - Symbian
  - uc/OS
  - RTEMS
  - T-Kernel
  - Integrity
  - ThreadX
- 国产嵌入式实时操作系统，如：
  - HOPEN
  - DeltaOS
  - SmartOS
  - SZOS
  - RT-Thread
  - DOOLOO RTOS

嵌入式实时操作系统－VxWorks

- VxWorks操作系统是美国WindRiver公司于1983年设计开发的嵌入式实时操作系统，具有高性能、稳定的内核以及友好的用户开发环境，是世界第一大嵌入式操作系统提供商，应用于航空航天、工业控制、网络设备、汽车电子等领域。
- 经典应用：1997年火星探路者、2007年凤凰号火星探测器、2012年好奇号火星探测车

在火星沙丘前进





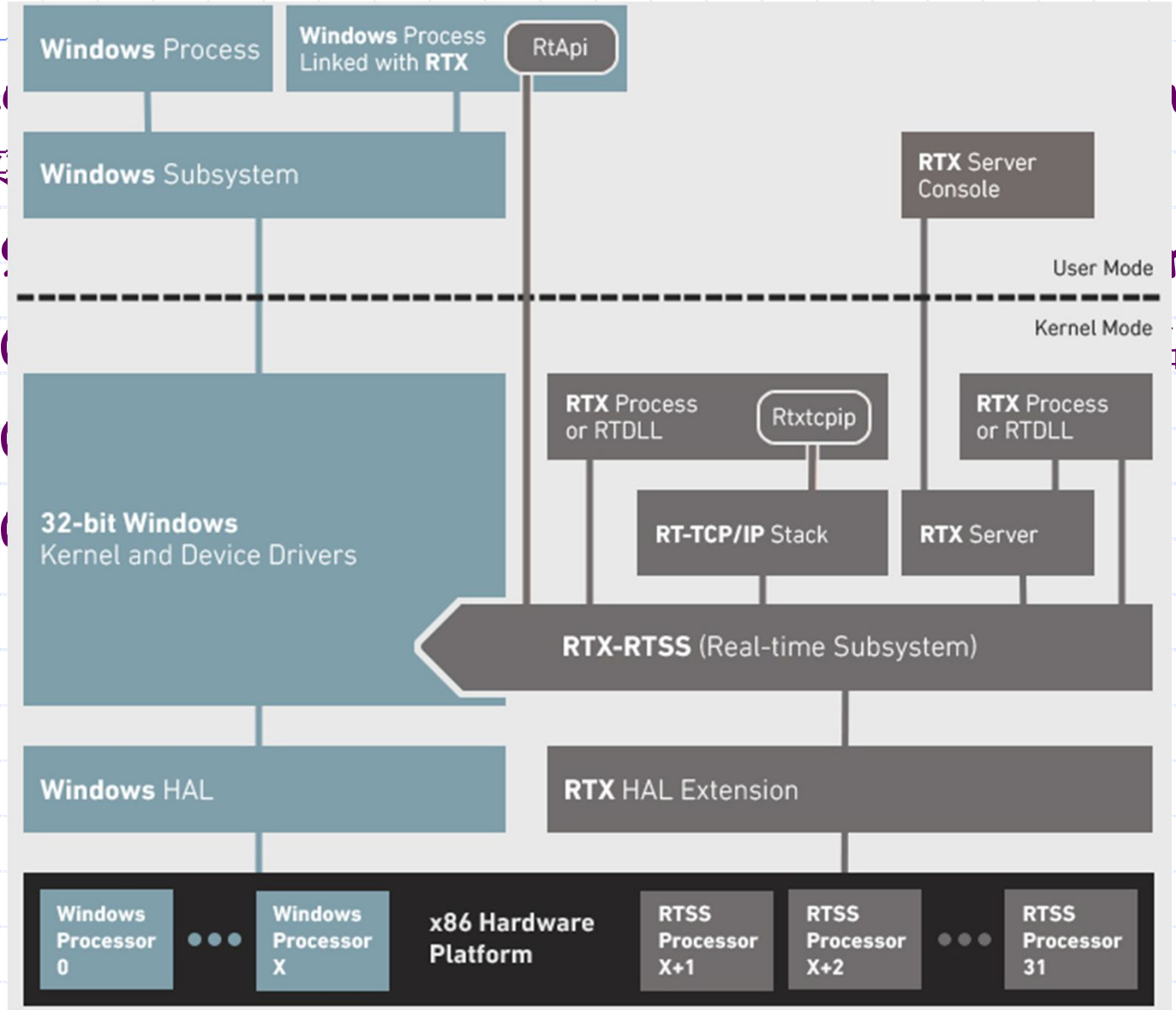
在火星上拍摄的日落全景

嵌入式实时操作系统－Integrity

- 美国Green Hills公司是世界排名第二的嵌入式操作系统提供商，Integrity是Green Hills公司的RTOS产品，代表了目前最先进的RTOS技术，被NASA JPL选中用于测试在太空中的新技术。
- 分为普通Embedded RTOS和关键应用中使用的DO-178B实时操作系统两类。
- 系统技术优势突出
  - 内核服务优化，系统调用的开销降至最小。
  - 复杂的系统调用可以被抢占。
  - 系统的调度器是一个真正的实时调度器。
  - 具有快速中断处理能力，内核从不阻塞某些中断。
  - 具有一流的集成开发环境MULTI®支持。

Windows实时化改造—RTX/RTX64

- R… …ws的
  实…
- 19… …n
- 20… …RTX
- 20…
- 20…



Linux实时化改造－实时Linux

- 嵌入式系统追求数字化、网络化和智能化，要求系统必须是开放的、提供标准的API，并能够方便地与众多第三方软硬件沟通。尤其是处于核心地位的操作系统。
- Linux是开放源码的，不存在黑箱技术，遍布全球的众多Linux爱好者是其开发的强大技术后盾。
- 对Linux进行实时性改造与裁剪，形成：
  - Clinux
  - Embedix
  - RTLinux
  - RTAI
  - Monta Vista Linux

开源嵌入式实时操作系统－μC/OS-II/III

- C/OS-II/III是一种基于优先级抢占式、可移植、可裁剪的多任务实时操作系统。绝大部分源码是用ANSI C写的，与硬件相关的那部分汇编代码被压缩至最低限度，使得系统移植性强。
- C/OSII诞生于90年代初，最初名称是 C/OS，由Jean J.Labrosse开发，并在网络上开源，其特点为短小、精悍。
- C/OSII经裁剪最小可达2KB，最小数据RAM需求10KB。
- C/OS-II/III可以在8位～64位，超过40种不同架构的微处理器上运行，在世界范围内得到广泛应用，包括：手机、路由器、集线器、不间断电源、飞行器、医疗设备及工业控制上。

开源的嵌入式实时操作系统－ThreadX

- ThreadX是一款强实时操作系统，以内核小（最小内核为2K，最小RAM 500byte）、实时性强、高可靠性、源代码开放，免收产品版权费而闻名。由美国Express Logic提供解决方案，适于深度嵌入的系统，有功能强大的开发调试环境MULTI®支持。
- 典型应用：2005年7月4日，美国NASA实施"深度撞击"号宇宙飞船对坦普尔1号彗星的准确撞击，关键任务由ThreadX完成。

开源的嵌入式实时操作系统－T-Kernel

- 由日本东京大学的坂村健教授主持开发，具有执行效率高、实时性好等特点。
- 1984年提出计算机操作系统规范TRON（The Real-time Operating system Nucleus）构想，先后推出了ITRON、JTRON、BTRON、CTRON等规范。
- 其应用从汽车、移动电话、传真机到电视机、家电等领域，主要用户包括：丰田、松下、日立、富士通、东芝、索尼、佳能、理光、NEC等，装机量超过30亿。
- IBM、Microsoft、ARM、MIPS、Sun、Oracle等企业相继加入其开放式系统架构。

开源的嵌入式实时操作系统－RT-Thread

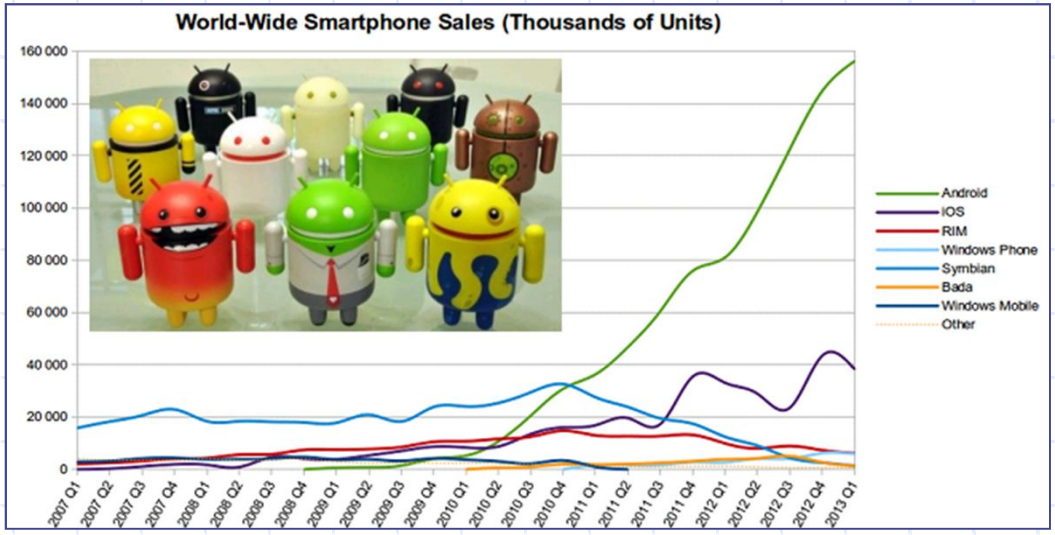
- DOOLOO RTOS开发人员，于2006年开发了RT-Thread基础内核，2015年2月发布2.0.0版本。
- RT-Thread获得2011年中日韩开源软件竞赛技术优胜奖，中国共三个（淘宝OceanBase、红旗Qomo Linux）。
- RT-Thread功能特点：小型、实时、可剪裁，核心能够小到4K ROM，1K RAM，支持精细裁剪。
- RT-Thread支持ARM7、ARM9、Cortex-M3、 M4、MIPS、AVR32、V850E，以及SH的16位MCU。

手机嵌入式操作系统－IPhone OS

- iPhone OS 或 OS X iPhone是由苹果公司为iPhone开发的操作系统
  - iPhone、iPod touch以及iPad
  - 以Darwin为基础
- 系统架构分为四个层次
  - 内核操作系统层（the Core OS layer）
  - 内核服务层（the Core Services layer）
  - 媒体层（the Media layer）
  - 可轻触层（the Cocoa Touch layer）
- 系统操作占用大概512MB的内存空间
- 源码模式：封闭源码+开放源码 组件
- 最新版本
  - 2015-3
  - iOS 8.2

手机嵌入式操作系统－Android

- Android是Google开发的基于Linux平台的开源手机嵌入式操作系统。
- Android的特点
  - 融入Web应用，如：Gmail、Google Maps、YouTube、Google日历、Google Talk
  - Android操作系统免费向开发人员提供
- 最新版本：2015年3月Android 5.1

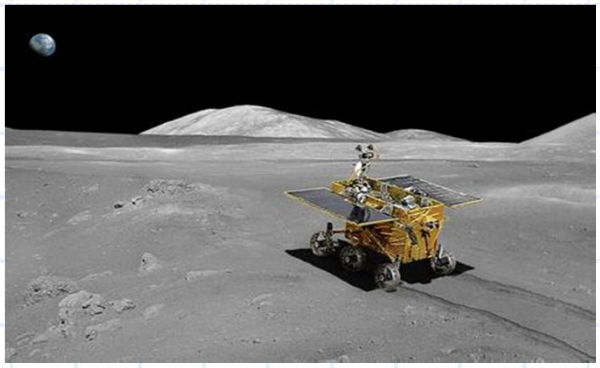


手机嵌入式操作系统－Windows phone

- 2010年2月，微软公司正式发布Windows Phone 7智能手机操作系统，Windows Mobile系列彻底退出了手机操作系统市场。
- 2011年2月，诺基亚在英国伦敦宣布与微软达成战略合作关系。诺基亚手机将采用Windows Phone系统，并且将参与系统开发。
- Windows phone把网络、个人电脑和手机的优势集于一身，提供良好的用户体验：
  - 仪表盘主屏
  - 桌面定制
  - 图标拖拽
  - 滑动控制
- 2012年10月发布了Window Phone 8

其它嵌入式操作系统

- 玉兔号操作系统——SpaceOS



- 机器人操作系统——ROS

课程大纲

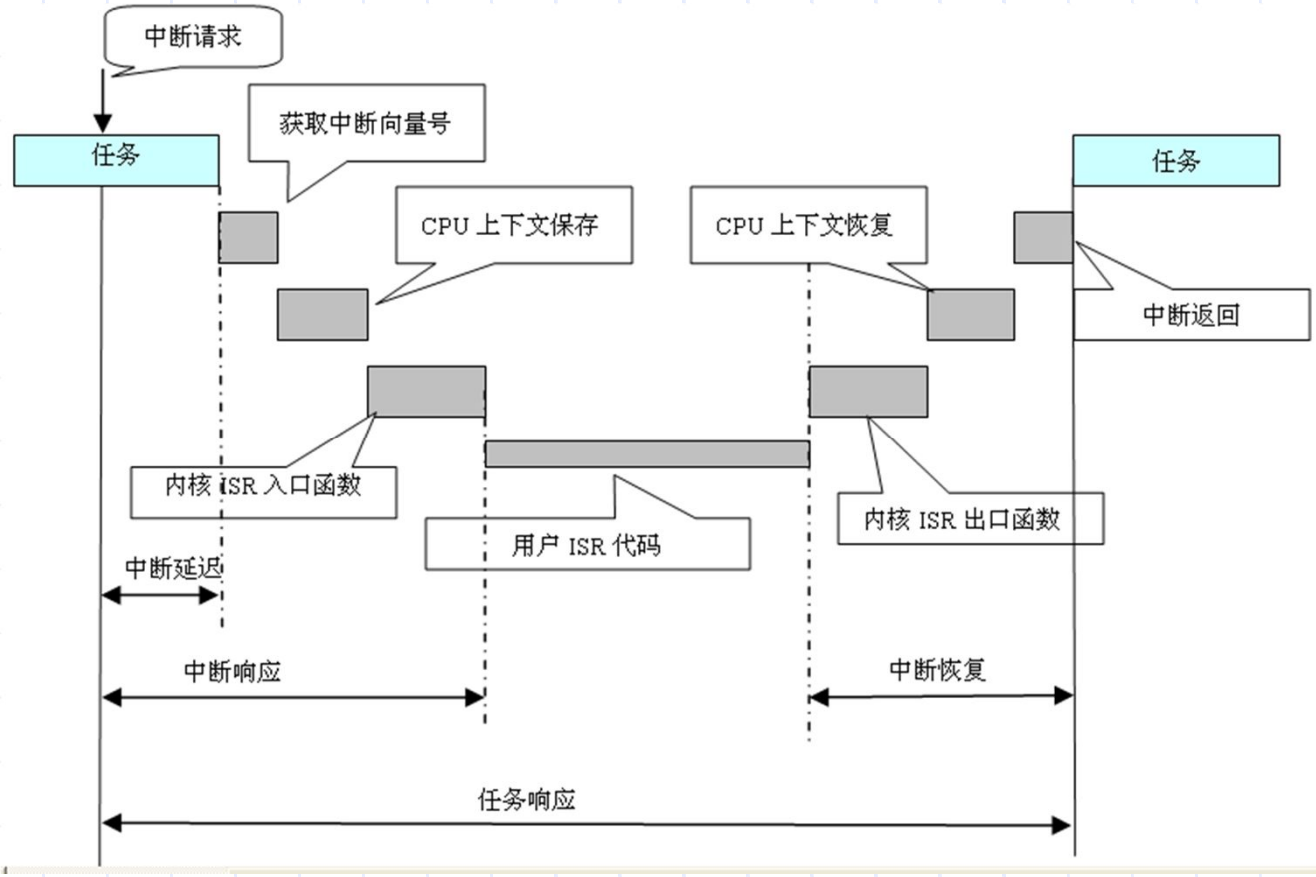
- 嵌入式实时操作系统概况
- 嵌入式实时操作系统特点
- 嵌入式实时操作系统功能简介

嵌入式实时操作系统内核重要特性

- 嵌入式实时操作系统内核的重要特性
  - 实时性
  - 可裁剪、可配置性
  - 可靠性支持
  - 应用编程接口支持
  - 可移植性

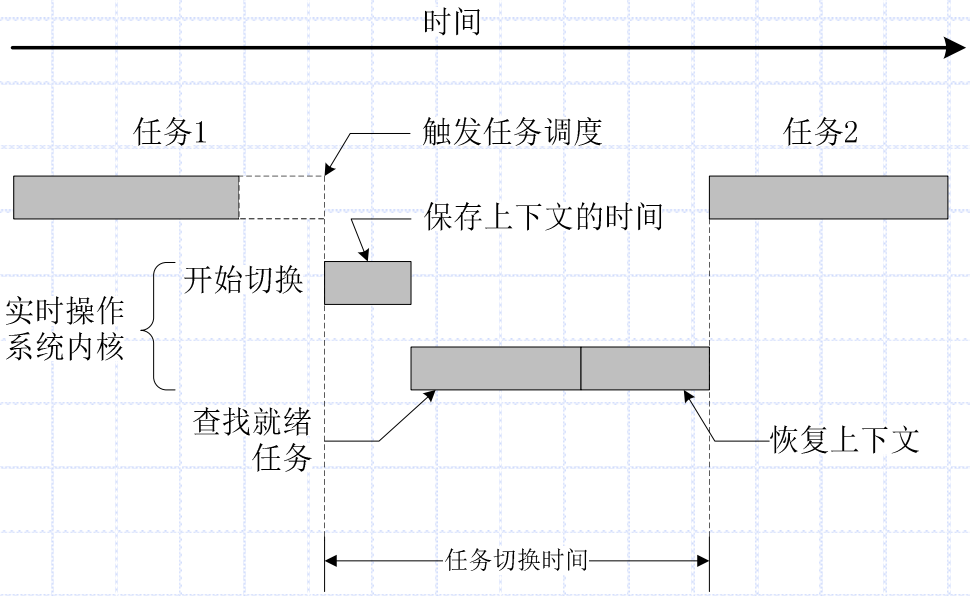
嵌入式实时操作系统内核实时性能指标

- 嵌入式实时操作系统内核的实时性能定量指标包括
  - 任务上下文切换时间
  - 中断延迟时间
  - 中断响应时间
  - 中断恢复时间
  - 任务响应时间



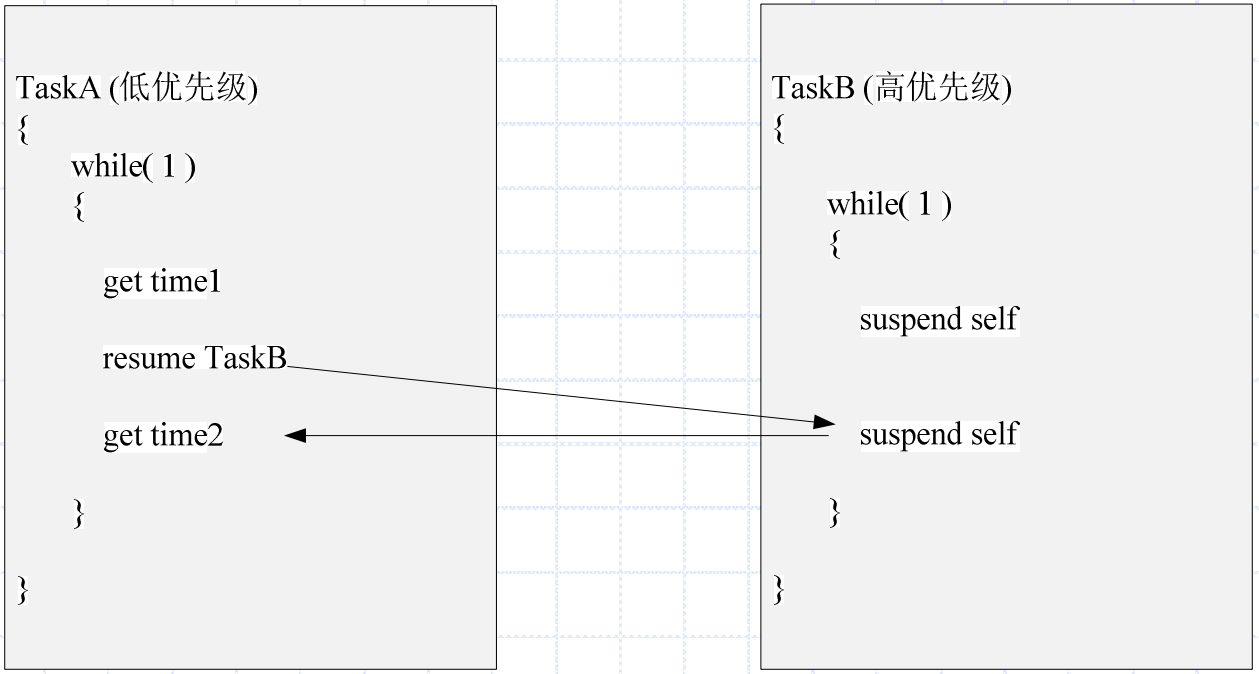
嵌入式实时操作系统内核实时性能关键指标

- 最大中断禁止时间
  - 反映内核对外界停止中断响应的最长时间
- 任务上下文切换时间
  - 系统中最频繁发生的动作，影响整个系统性能
  - 包括：保存当前任务上下文、选择新任务、恢复新任务上下文



任务上下文切换时间——两次悬挂法（1/2）

- 任务上下文切换时间测试



$$T_1 = t_2 - t_1 = t_{taskresume} + t_{tasksuspend} + 2 * t_{ctxsw}$$

任务上下文切换时间——两次悬挂法（2/2）

- 任务上下文切换时间测试

```
TaskA (低优先级)
{
    while( 1 )
    {
      //do something
      //do something
      //do something
    }
}
```

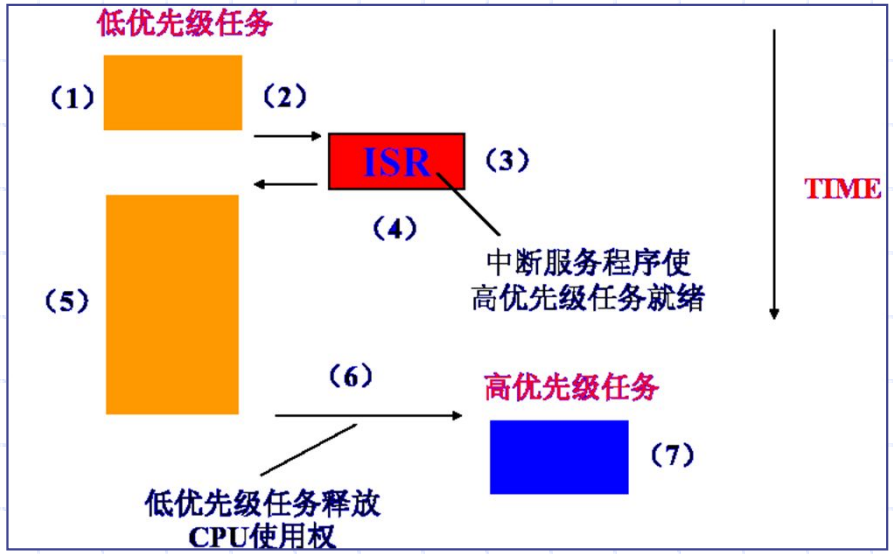
```
TaskB (高优先级)
{
    while( 1 )
    {
      get time1
      suspend TaskA
      resume TaskA
      get time2
    }
}
```

$$T_2 = t_2 - t_1 = t_{taskresume} + t_{tasksuspend}$$

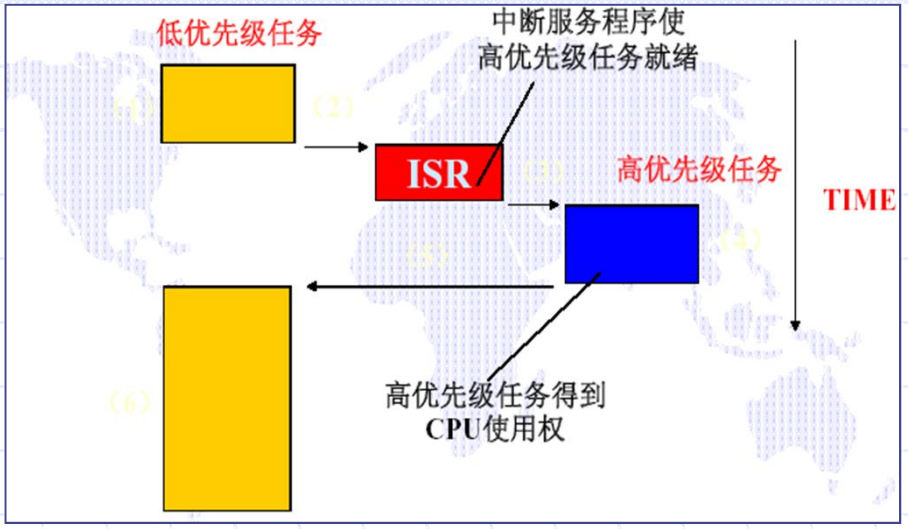
$$t_{ctxsw} = (T_1 - T_2)/2$$

提高内核实时性的方法－任务调度算法

- 通用操作系统－非抢占式调度
  - 公平和最小化任务平均响应时间
  - 提高系统吞吐率

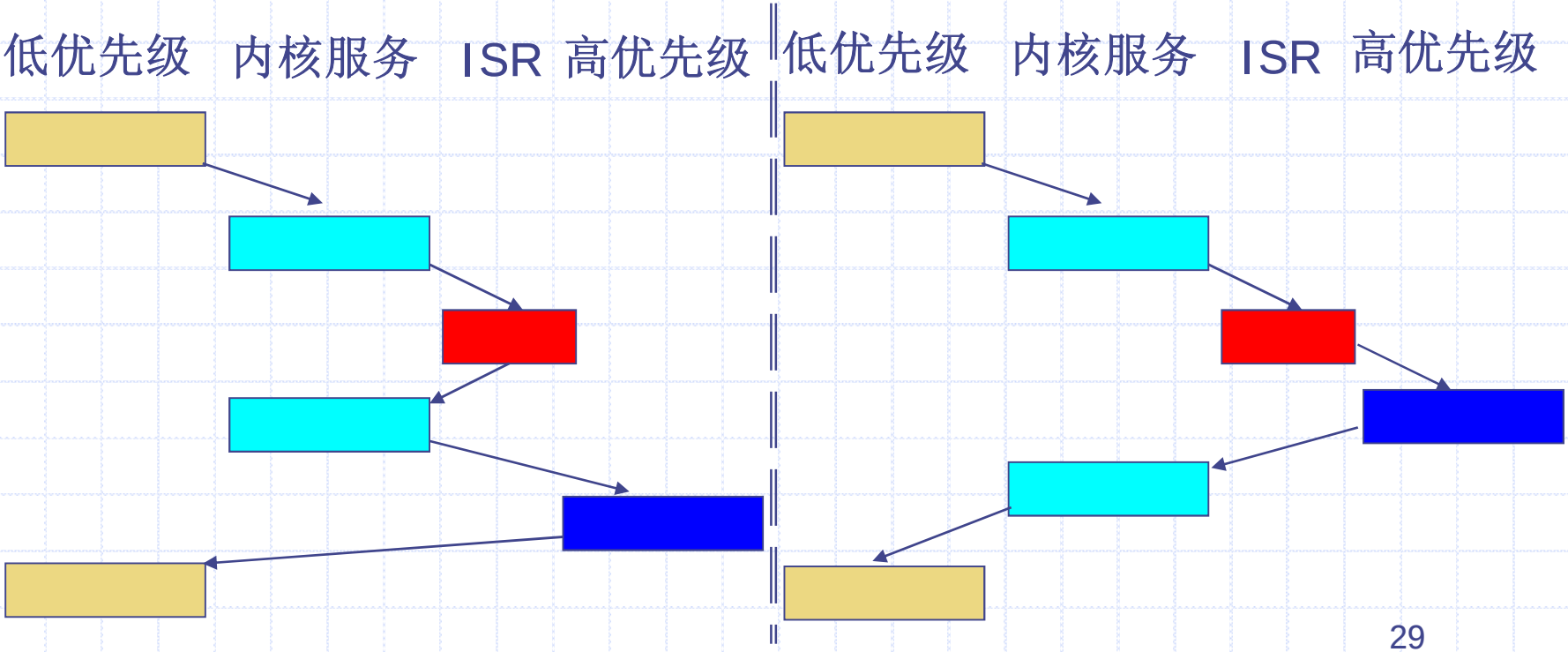


- 嵌入式实时操作系统－抢占式调度
  - 提高对关键性任务响应
  - 关注最坏执行时间
  - 函数的可重入性设计



提高内核实时性的方法－可抢占内核

- 通用操作系统－不可抢占内核
  - 内核服务不能被中断
  - 内核服务可中断，但不调度
- 嵌入式实时操作系统－可抢占内核
  - 内核服务可响应中断
  - 中断退出后可进行调度



提高内核实时性的方法－内核关中断时间

- 通用

三种RTOS的系统调用对比分析

uC/OSII 系统调用代码示例

```
OSTaskResume
{
  ……
  //关闭中断
  OS_ENTER_CRITICAL();

  //访问内核数据结构，完成系统调用处理
  ……

  //开启中断
  OS_EXIT_CRITICAL();
  ……
}
```

巨型内核锁模型

RTEMS 系统调用代码示例

```
_Thread_Resume
{
  ……
  //关闭中断
  _ISR_Disable();
  //访问内核数据结构，完成部分系统调用处理
  ……
  //设置内核抢占点，可响应外部中断
  _ISR_Flash();
  //继续完成系统调用处理
  ……
  //开启中断
  _ISR_Enable();
  ……
}
```

内核抢占点模型

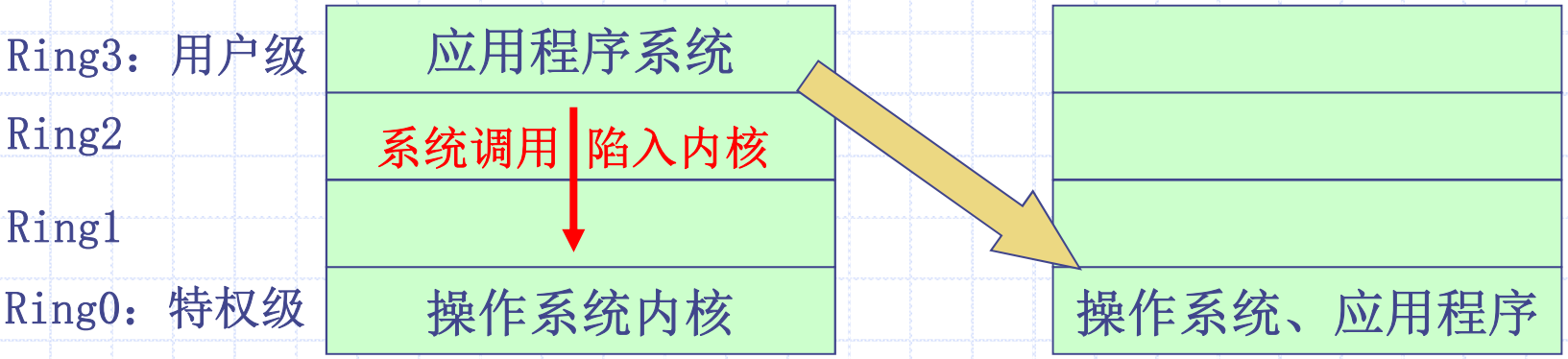
VxWorks 系统调用代码示例

```
taskResume
{
  ……
  //是否处于内核态
  if(kernelLock)
  {
   //已处于内核态，将系统调用处理加入工作队列后退出
     workQAdd(call);
   }
   else //未处于内核态
   {
    //设置内核态标记
     kernelState=True;
    //完成系统调用处理
     vxSysCall();
   }
}
```

基于延迟工作队列的内核可重入模型

提高内核实时性的方法－系统运行状态

- 许多嵌入式操作系统不划分“系统空间”和“用户空间”，如VxWorks、RTEMS等，操作系统内核与外围应用程序之间不再有物理的边界，系统中“进程”实际上都是内核线程。
- 操作系统、应用程序均运行在特权级别的优缺点：
  - 优点：减少由于空间切换导致的执行开销，提高实时性。
  - 缺陷：应用程序可破坏操作系统内核，导致系统崩溃。



通用系统运行状态　　实时系统运行状态

提高内核实时性的方法－存储管理机制

- 不支持虚拟存储：如果采用虚存技术，一个实时任务执行的最坏情况是每次访存都需要调页，如此累计起来的该任务在最坏情况下的运行时间是不可预测的，因此实时性无法得到保证。许多嵌入式操作系统不直接支持虚拟存储管理技术。
- 不支持动态内存分配：由于动态内存分配具有时间及分配结果的不确定性，因而在强实时型系统（OSEK）中采用静态内存分配方法，即在系统初始化时，为每个实时任务划分固定的内存区域，系统运行只使用内存，而不再分配内存和释放内存。

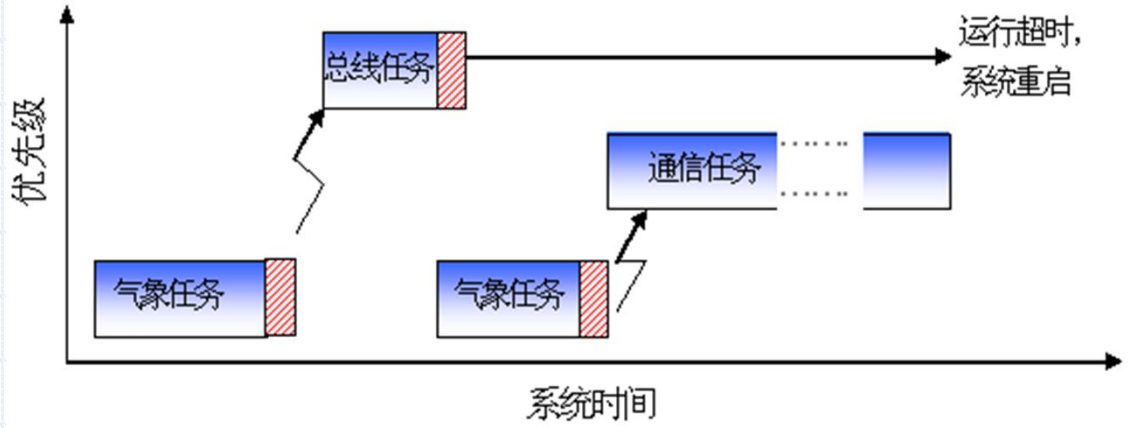
提高内核实时性的方法—任务互斥、同步

- 资源有限等待：任务没能获得需要的资源会被阻塞。如果资源不是任务继续运行必备的，任务可选择有限等待该资源。
- 优先级逆转问题解决—抢占式任务调度中的资源竞争：

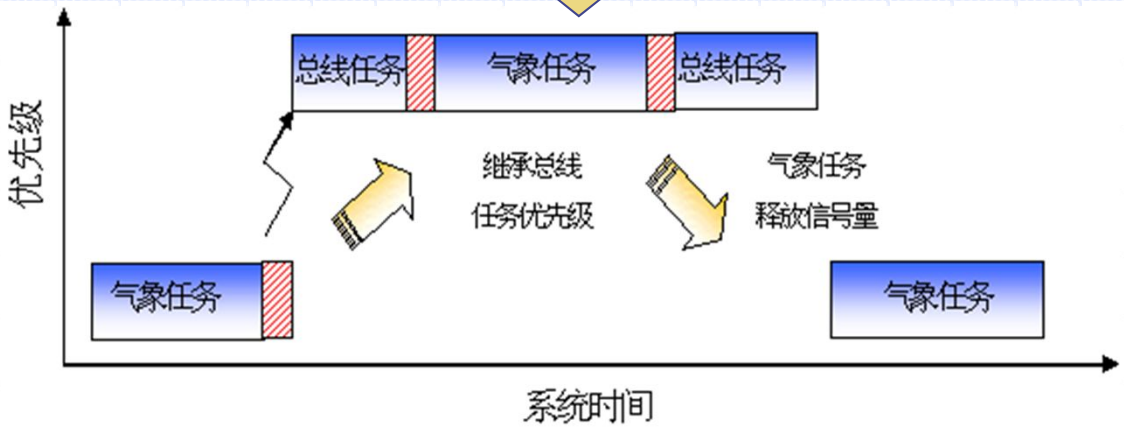
- 1997年7月4日，火星探路者在火星表面成功着陆并进行观测，发回了各种火星表面全景图，被大肆宣称为“完美”。但是在着陆后的第10天，也就是开始采集气象资料后不久，探路者开始犯傻，反复无规律地重启，每次重启都造成了数据丢失，在每天的记者招待会上这都是记者们不会放过的最热门的话题。
- JPL（美国国家航空航天局喷气推进实验室）的工程师们花了相当多的时间在实验室仿真，希望能够再现引起重启的情况。几天过去了，一个清晨，几乎所有的工程师都走了，只剩下最后一位Mr. So-So的时候，火星上那台探路者兄弟身上发生的重启情况终于被再现了。经过数据分析，得出了原因——优先级逆转。

提高内核实时性的方法——优先级逆转问题

- 嵌入式实时操作系统——优先级逆转现象

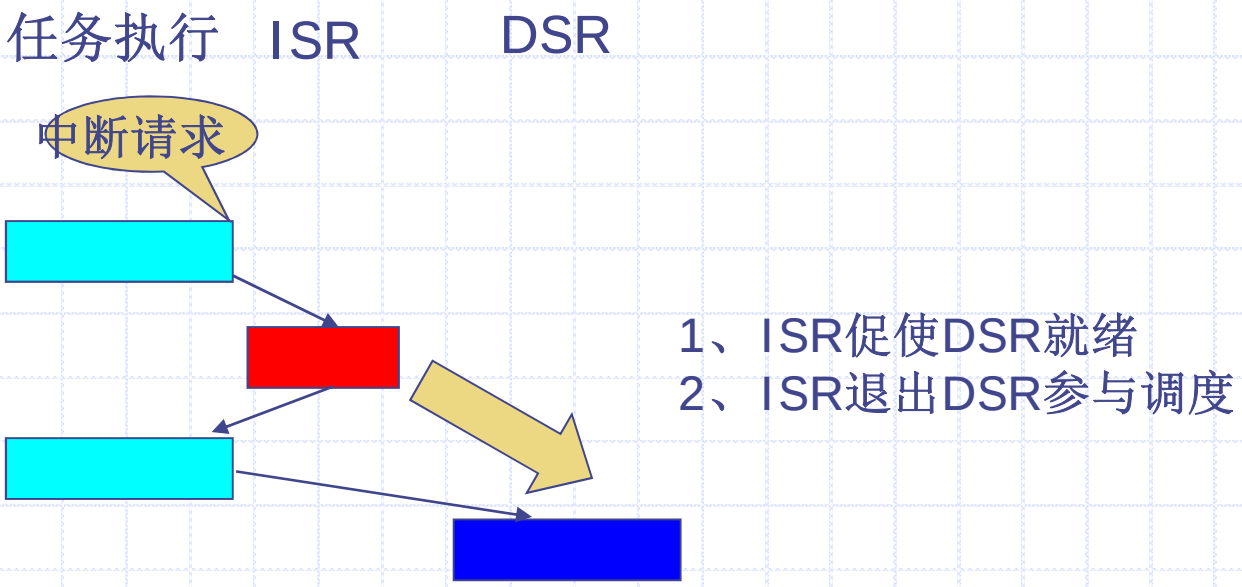


采用优先级继承协议消除



提高内核实时性的方法－中断处理

- 中断嵌套处理：确保高优先级的中断能及时处理。
- 中断服务层次化：对中断的处理，不需要完全由中断服务程序（ISR）进行处理，采用ISR与任务相结合的方法处理，如eCos系统，分为两个层次进行：ISR、中断滞后服务程序DSR。ISR在响应中断时立即调用，DSR由ISR发出请求后调用。



嵌入式实时操作系统内核的可裁剪、可配置性

- 可裁剪性：用以满足不同复杂程度的应用需求。嵌入式环境资源配置及需求情况各异，一般只要求嵌入式操作系统的功能子集，因而需要裁剪掉部分功能，并保证功能的相对完整性。内核的可裁剪程度取决与模块之间的耦合程度。
- 裁剪方法：模块级裁剪、函数级裁剪、代码级裁剪
- 一个最小的多任务嵌入式软件包括：
  - Bootloader
  - 具有任务管理及定时功能的基本内核
  - 一个初始化任务
- 可配置性：可根据应用需求，配置系统任务数目、调度算法、任务堆栈等。

嵌入式实时操作系统内核的可靠性

- 可靠性对于实时系统比非实时应用系统更为重要。
- 嵌入式实时操作系统内核提供诸多机制进行保障：异步信号、定时器、异常处理、用户扩展、内存保护等。
- 典型内核可靠性增强技术：
  - 内存释放清理
  - 冗余内存分配
  - 内存冗余编码
  - 内存保护增强
  - 看门狗支持增强

嵌入式实时操作系统内核的应用编程接口

- 每一个嵌入式操作系统提供的应用编程接口（系统调用）的功能和种类都不相同，种类越多、功能越强越好。
- 应用编程接口的标准化：
  - POSIX(a Portable Operating System Interface based on Unix)实时系统标准，POSIX1003.1c、1003.1d
  - 汽车电子标准：OSEK
  - 航空电子标准：ARINC653（APEX接口）
  - 电气电子标准：IEC61508
  - 信息家电规范：T-Kernel

嵌入式实时操作系统的安全性认证

- EAL/CC：CC安全评估是1999年起效的一项国际安全标准，共分为7级安全评估。VxWorks、Integrity均通过了EAL6+认证。

- DO-178B/ED-12B：美国航空无线电技术委员会（RTCA）提出，被美国联邦航空局/欧洲航空管理部门接受的机载软件适航认证。VxWorks、Integrity、 C/OSII均得到Level A认证。

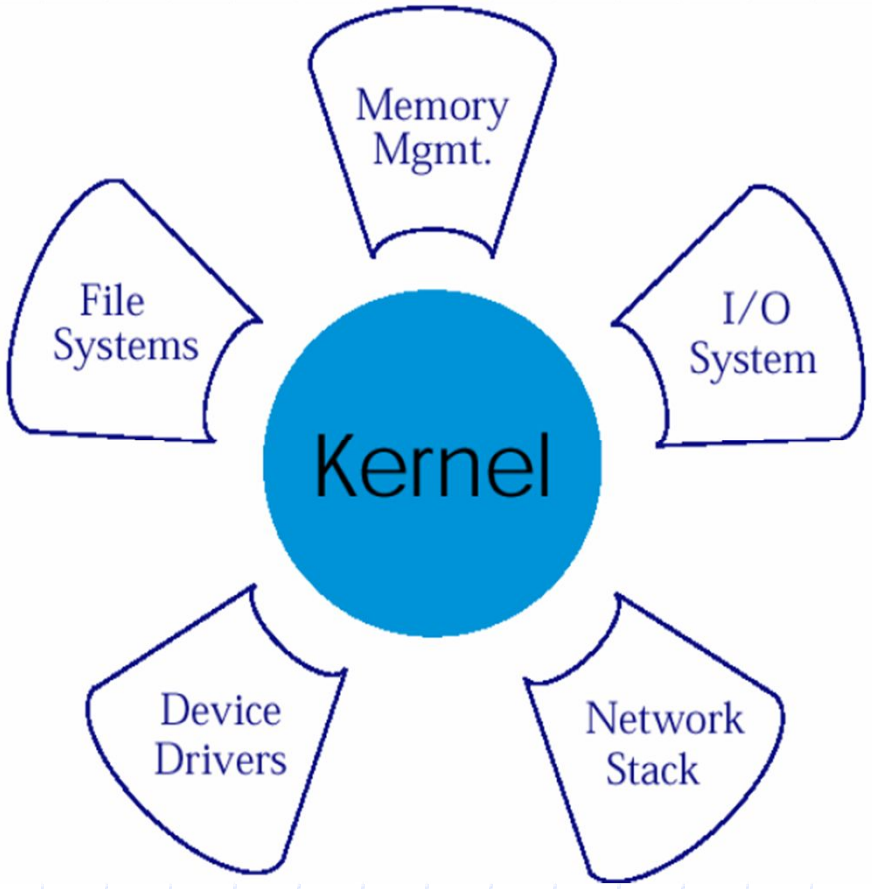
- OSEK/VDX：欧共体汽车产业联盟规定的汽车电子嵌入式系统标准。风河的MotoWorks、微软的Windows Automotive、Nucleus OSEK、OSEKturbo均得到认证。

课程大纲

- 嵌入式实时操作系统概况
- 嵌入式实时操作系统特点
- 嵌入式实时操作系统功能简介

嵌入式实时操作系统内核基本功能

- 嵌入式实时操作系统内核的基本功能
  - 实时多任务管理
  - 中断与异常管理
  - 共享资源互斥管理
  - 多任务同步与互斥
  - 内存管理
  - 时钟定时器管理
  - 电源管理

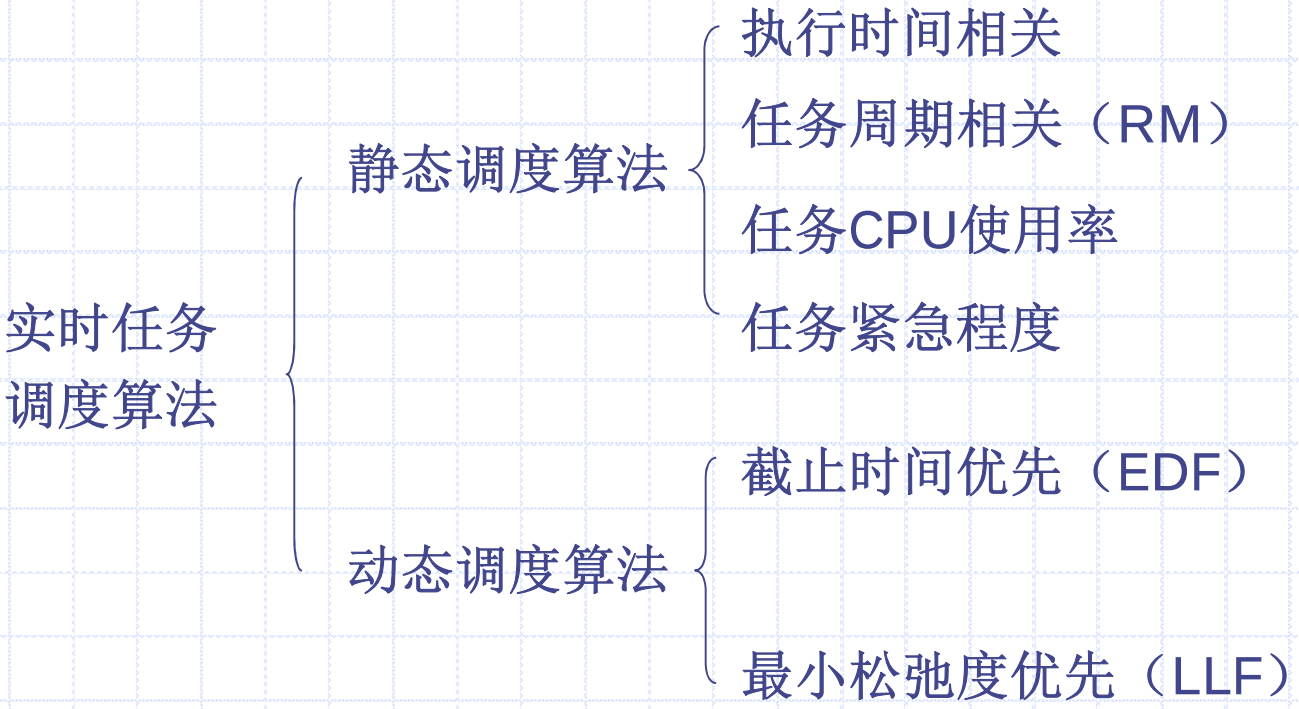


实时内核基本功能－任务调度

- 1970年，美国UIUC大学的C.Liu、Jane教授建立了RTSL（real time system lab）实验室。
- 1973年，C.Liu、Layland在ACM杂志上，提出并分析了单调速率调度算法（Rate Monotonic，RM）和时限调度算法（Deadline），开辟了实时系统抢占式任务调度算法、可调度性分析领域的先河。

实时内核基本功能－实时任务调度算法分类

- 在实时任务抢占式调度算法中，根据任务的优先级确定时机，实时任务调度算法可分为静态调度和动态调度两类。



实时内核基本功能—任务调度经典算法举例

- 单调速率调度算法（C.Liu、Layland；ACM，1973 ）
  - ✓ 现代实时系统任务调度的理论基础
  - ✓ 最佳的静态调度算法
  - ✓ 算法建立在下述假设基础上
    - 所有任务都是周期任务
    - 每个任务执行截止期等于该任务的周期
    - 每个任务在周期中，执行时间固定，保持常量
    - 任务之间不通信，也不同步
    - 任务可以在任何位置被抢占，不存在临界区

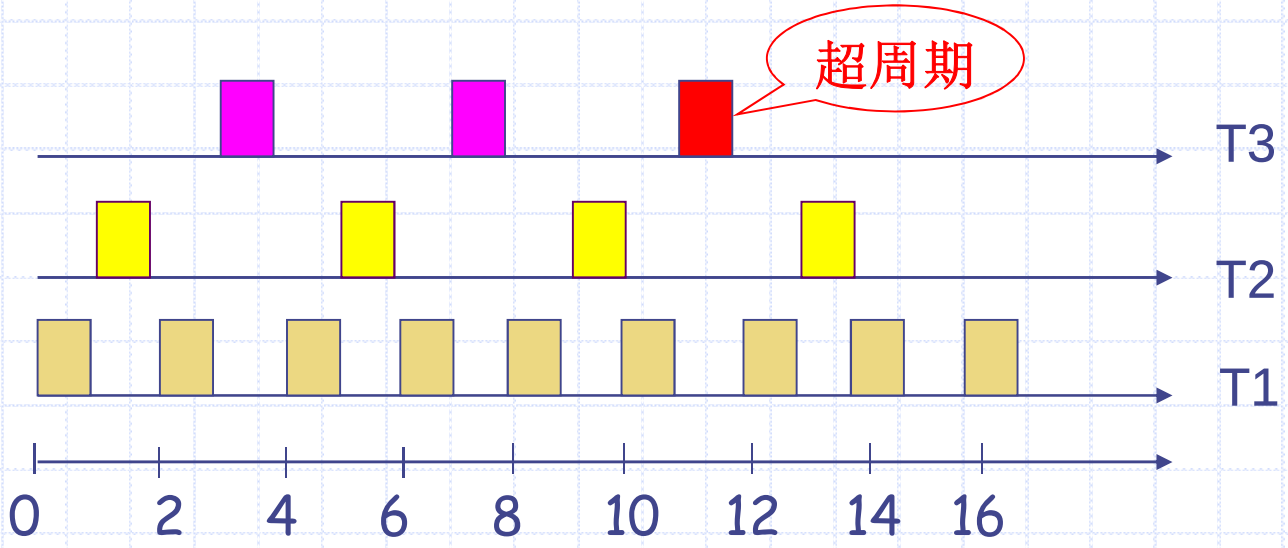
不可调度：指某一个任务在周期内无法完成任务，即：

任务的执行结束时间 > 任务的截止期

实时内核基本功能—任务调度经典算法举例

- 不可调度情况举例
- 假设系统存在任务、执行时间及运行周期如下

任务	执行时间	周期	优先级
T1	1	2	1
T2	1	4	2
T3	3	8	3



实时内核基本功能—任务调度经典算法举例

- RM算法规定：任务的优先级与任务的周期表现为单调函数，任务周期越短，优先级越高。
- 对RM算法研究的贡献在于
  - ✓ 提出了临界时间概念，用于判定调度过程中的最坏情况；
  - ✓ 证明了RM算法是静态调度算法中的最优性；
  - ✓ 提出了一个RM算法中任务可调度性分析的充分条件。
- 临界时间：一个任务响应所需的最大时间称为临界时间。
- 如果所有任务的临界时间均小于任务周期，则任务可调度。

  一个任务什么时候到达其临界时间？
- 定理：任何任务在与比其优先级高的所有任务同时被触发时，将达到其临界时间。

实时内核基本功能－任务调度经典算法举例

- 定理：如果一个任务集能够被其他静态算法调度，那么RM算法就一定能调度这个任务集，即RM调度是最优的静态调度算法。
- 证明：交换法
  - ✓ 假设一个任务集S采用其他静态优先级算法可以调度，设ti和tj是其中两个优先级相邻的任务，Ti > Tj，而Pi < Pj（即长周期任务的优先级高），将ti和tj的优先级互换，可以证明这时S仍然可以调度：
    - 交换这两个任务优先级，不会影响其它任务的完成时间；
    - Tj执行时间提前，因而必定不会超过截止时间；
    - Ti的执行时间＝高优先级任务的执行时间＋tj执行时间＋ti执行时间 < Tj < Ti，因而，Ti执行也不会超过截止时间。
  - ✓ 按照上述交换方法，任何静态优先级调度最终都可以转换成RM调度。

实时内核基本功能—任务调度经典算法举例

- RM算法中任务可调度性分析的一个充分条件：

$$\sum_{i=1}^{n}\frac{C}{T} \le n\times(2^{\frac{1}{n}}-1)$$

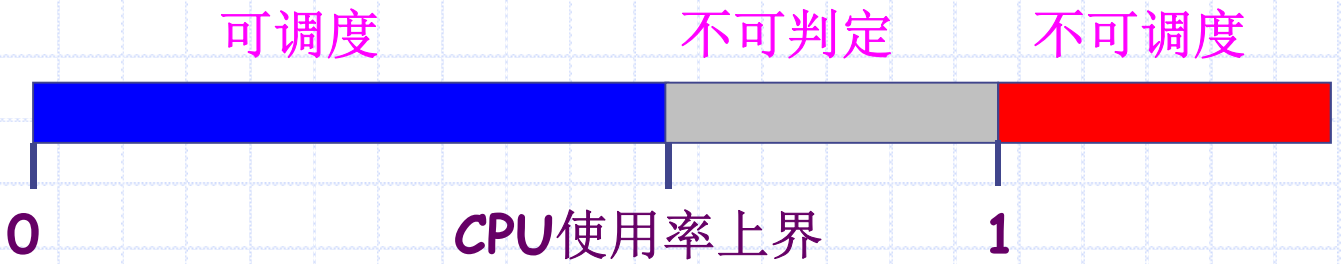
CPU使用率上界

其中，C为任务执行时间，T为任务周期

任务数量	可调度的CPU使用率	任务数量	可调度的CPU使用率
1	1	5	0.743
2	0.828	6	0.735
3	0.780	……	……
4	0.757	∞	ln2≈0.6931

实时内核基本功能—任务调度经典算法举例

- 调度可判定性物理意义：



任务	执行时间	周期	优先级
T1	1	2	1
T2	1	4	2
T3	3	8	3

- 可调度性判定举例：

1/2+1/4+3/8 = 1.125 > 1 > 0.780，不可调度！

实时内核基本功能－中断管理

- 中断是一种异步机制，中断服务程序（ISR）不需要内核的调度就可以执行。
- 但ISR要和其他应用任务之间协作，以快速、合理响应外部事件。
- 内核提供与中断相关的功能：
  - 挂接ISR：中断向量与处理函数关联
  - 获取ISR入口地址
  - 获取中断嵌套层数
  - 开/关中断

实时内核基本功能－中断管理

中断服务程序代码示例

```
__interrupt double compute_area (double radius)
{
double area = PI * radius * radius;
printf(" Area = %f", area);

return area;
}
```

- 不能传递参数
- 不能提供返回值
- 不能调用printf函数
- 尽量不要在ISR中进行浮点运算

实时内核基本功能－中断管理

- 中断服务程序设计中需特别注意中断冲突问题：
  - 当ISR、ISR之间，或ISR、任务之间共享变量，或调用含有共享变量的函数时，需防止共享变量冲突；
  - 当ISR、ISR之间，或ISR、任务之间共享寄存器，或调用含有共享寄存器的函数时，需防止寄存器冲突。
  - ISR不允许执行I/O操作，或调用含有I/O操作的函数。
  - ISR不允许申请信号量（但可以释放信号量！），或调用含有申请信号量操作的函数（如malloc）。

实时内核基本功能—共享资源互斥

- 实现共享资源互斥的方法很多，不同之处在于互斥的影响范围和程度不同，常用的方法包括：
  - 关中断：互斥力度最强，但可能降低系统实时性
  - 测试并置位指令：利用某个全局变量判断资源互斥

```
lock = 0; key = 1;
……
_asm(" xchg(&lock, &key) ");
if(key == 0)
  进入临界区代码 ;
```

  - 禁止任务抢占：对任务调度上锁，但不禁止中断

```
OSSchedLock();
……
OSSchedUnlock();
```

  - 使用信号量：对共享资源上锁，比关中断、禁止任务抢占粒度更精细

实时内核基本功能—共享资源互斥方法比较

	关中断	测试并置位	禁止抢占	信号量
锁定范围	所有可屏蔽中断事件	所有使用该指令的代码	所有任务	竞争共享资源的任务
响应时间影响	如果时间长，影响较大	较小	如果时间长，影响较大	可能导致优先级反转
系统开销	小	小	小	较大
注意事项	时间尽量短	可能影响可移植性	时间尽量短	避免过度使用

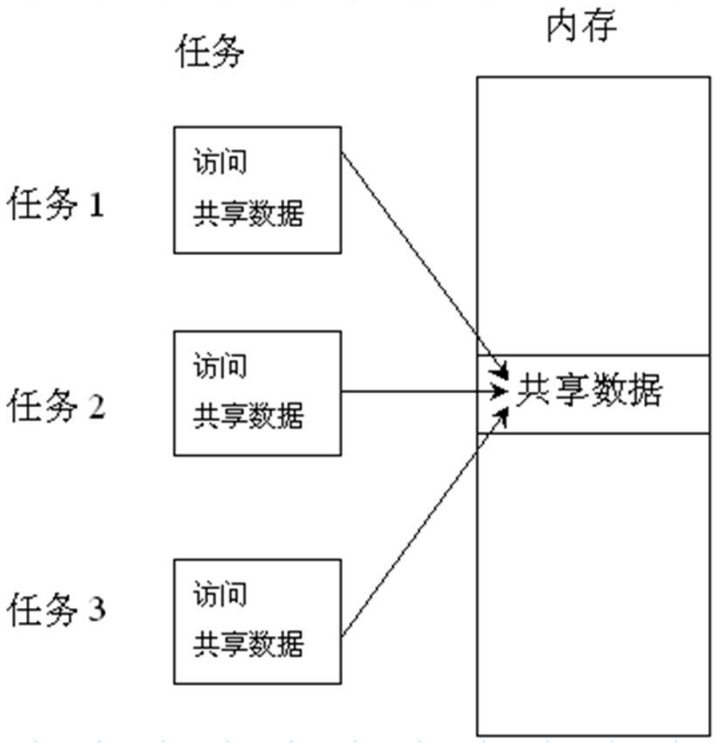
- 共享资源互斥的设计原则：
  - 当任务之间互斥，可使用所有方法，测试/置位、信号量方法，对其他任务运行的干扰小；
  - 当ISR之间互斥，只能使用关中断法；
  - 当ISR与任务之间互斥，只能使用关中断法。

实时内核基本功能—同步与通讯

- 同步与通讯的需求
  - ✓ 任务~任务之间：单向、双向
  - ✓ **ISR**~任务之间：单向
- 常用的同步、通讯机制：
  - ✓ 共享内存
  - ✓ 信号量
  - ✓ 消息：邮箱、消息队列
  - ✓ 事件
  - ✓ 信号
  - ✓ 管道

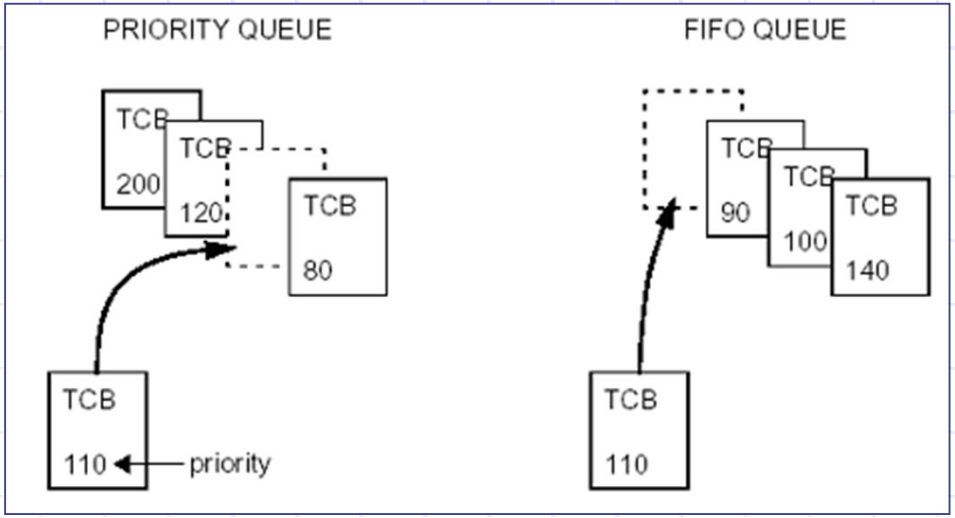
实时内核基本功能—通讯

- 共享数据结构
  - ✓ 最直接的任务间通信方式
  - ✓ 全局变量、线性缓冲区、循环缓冲区、链表，可以被不同上下文环境中运行的代码直接访问
  - ✓ 需采用互斥方法进行保护



实时内核基本功能—同步、互斥

- 信号量：解决任务间同步与互斥的主要手段。
- 常用信号量分类
  - ✓ 二元信号量（**binary**）：快速、通用，对互斥与同步做了优化。
  - ✓ 互斥信号量（**mutex**）：针对互斥问题进行优化的二元信号量。
    - 递归资源访问：如递归调用包含获取信号量的函数体
    - 安全删除问题：已获取信号量的任务不被意外删除
  - ✓ 计数信号量（**counting**）：控制共享资源的多个实例。
- 被信号量阻塞的任务排队策略
  - ✓ **FIFO**
  - ✓ 优先级排序



实时内核基本功能—通讯

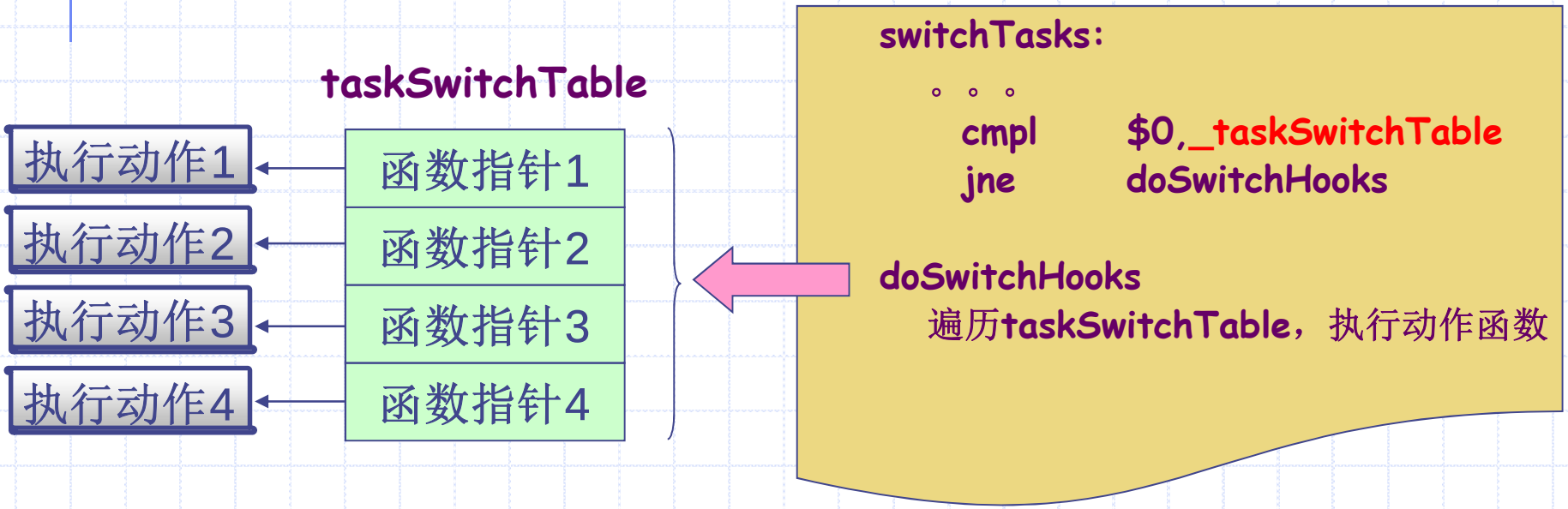
- 消息
  - ✓ 是内存空间中一段长度可变的缓冲区。
  - ✓ 是一种在任务之间、 ISR ～任务之间的通讯机制，注意：ISR只可以写消息，但不能读消息！
- 常用消息分类：
  - ✓ 邮箱（mailbox）：传递简单消息
  - ✓ 消息队列（message queue）：传递可变长的复杂消息
    - 消息进入队列的策略
      - FIFO
      - 优先级排序

实时内核基本功能－同步与通讯

- 管道
  - ✓ 管道是一个虚设备，提供了通过I/O设备接口访问消息队列的一个界面。任务可以使用标准的I/O接口open、read、write，以及ioctl调用。
- 事件
  - ✓ 用于实现任务之间、ISR～任务之间多对一、多对多的同步操作，通讯数据量小，主要动作分为接收事件、发送事件。
- 信号
  - ✓ 用于实现任务之间、ISR～任务之间的异步操作。

实时内核基本功能－用户扩展管理

- 在不更改内核代码的情况下，在内核调用点扩展用户功能。
- 内核可提供的扩展点包括：
  - ✓ 任务创建、任务启动、任务删除、任务上下文切换、任务退出
- 例如：在任务上下文切换时扩展增加功能



谢 谢！