嵌入式系统

北京邮电大学
计算机学院

戴志涛

嵌入式系统的软件

- 嵌入式系统软件的程序结构
- 硬件抽象层（HAL）与板级支持包（BSP）
- 引导加载程序（bootloader）
- 嵌入式系统的设备驱动程序
- 实时系统与实时操作系统

嵌入式系统软件的程序结构

- 简单循环轮询（Round-Robin）结构
- 带中断的循环轮询（Round-Robin）结构
  - 中断驱动结构/前后台结构
- 简单监控式的嵌入式操作系统结构
- 高性能通用嵌入式操作系统结构

嵌入式系统软件的程序结构

简单循环轮询（Round-Robin）结构

- 程序重复循环检查每个外部输入条件，一旦有需要处理的任务，则进行相应的处理
- 通常的软件结构：

```
初始化；
while（TRUE）{
        if（I/O设备1需要服务） 执行设备1服务函数；
        if（I/O设备2需要服务） 执行设备2服务函数；
        if（I/O设备3需要服务） 执行设备3服务函数；
        ……
        if（I/O设备n需要服务） 执行设备n服务函数；
}
```

嵌入式系统软件的程序结构

简单循环轮询（Round-Robin）结构

- 优点：
  - 程序结构简单，便于编程
  - 没有中断机制，不会出现程序随机切换带来的潜在问题
- 只适用于系统的任务数量较少、任务处理简单，且实时性要求不高的场景

嵌入式系统软件的程序结构

- 带中断的循环轮询（Round-Robin）结构
  - 在简单循环轮询结构的“主循环”基础上增加中断服务程序
  - 中断服务程序处理特别紧急的服务请求，然后设置状态标志；循环主程序轮询这些状态标志，并进行后续的处理
  - 中断驱动结构/前后台结构
    - 后台：循环执行的轮询程序
    - 前台：由若干中断服务程序组成
    - 当有外部事件发生时，外部事件提出中断请求，暂停后台运行的主循环，进行前台处理，处理完成后又回到后台继续运行主循环

嵌入式系统软件的程序结构

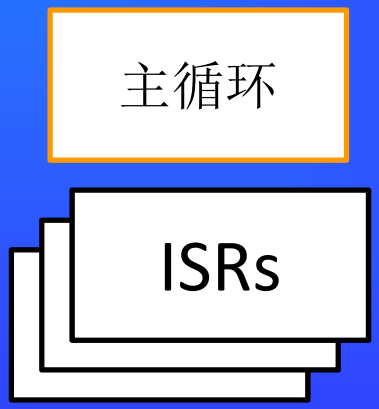
- 带中断的循环轮询结构/中断驱动结构/前后台结构
  - 通常的软件结构：

```
BOOL SA = FALSE;    /* 事件A状态标志初值 */
BOOL SB = FALSE;    /* 事件B状态标志初值 */
……
事件A中断服务程序
{
      设备A紧急操作;
      SA=TRUE;
}
事件B中断服务程序
{
      事件B紧急操作;
      SB=TRUE;
}
……
```

```
void main(void)    /* 后台循环程序 */
{
    while (TRUE) {
        if (SA) {
            SA = FALSE;
            事件A非紧急操作}
        if (SB) {
            SB = FALSE;
            事件B非紧急操作}
        ……
    }
}
```

嵌入式系统软件的程序结构

- 带中断的循环轮询结构/中断驱动结构/前后台结构
  - 提高了系统对紧急事件的响应速度，可以并发处理不同的异步事件
  - 由于中断的引入，使得系统软件复杂度明显提高
    - 必须谨慎处理中断嵌套、中断服务程序与主程序的同步协调等问题



嵌入式系统软件的程序结构

- **简单监控式的嵌入式操作系统结构**
  - **选配嵌入式操作系统（Embedded Operating System，EOS）**
    - 保证程序执行的实时性、可靠性和可维护性
    - 减少开发时间，降低系统的复杂性，保障软件质量
  - **小型的监控式嵌入式操作系统通常只包含内核（kernal），仅实现任务调度、任务间通信和中断管理等最基本功能，以多个并发的任务（task）取代“主循环”**
  - **操作系统内核负责多任务处理，执行任务创建与初始化、任务调度、存储管理、时钟管理和中断管理等**

嵌入式系统软件的程序结构

简单监控式的嵌入式操作系统结构

简单监控式的嵌入式操作系统结构：

```
事件A中断服务程序
{
        事件A紧急操作;
        发送信号S;
}
事件B中断服务程序
{
        事件B紧急操作;
        发送信号T;
}
……
```

```
void  task1(void)
{
    while  (TRUE)  {
            等待信号S;
            事件A非紧急操作;
            }
}
void  task2(void)
{
    while  (TRUE)  {
            等待信号T;
            事件B非紧急操作;
            }
}
……
```

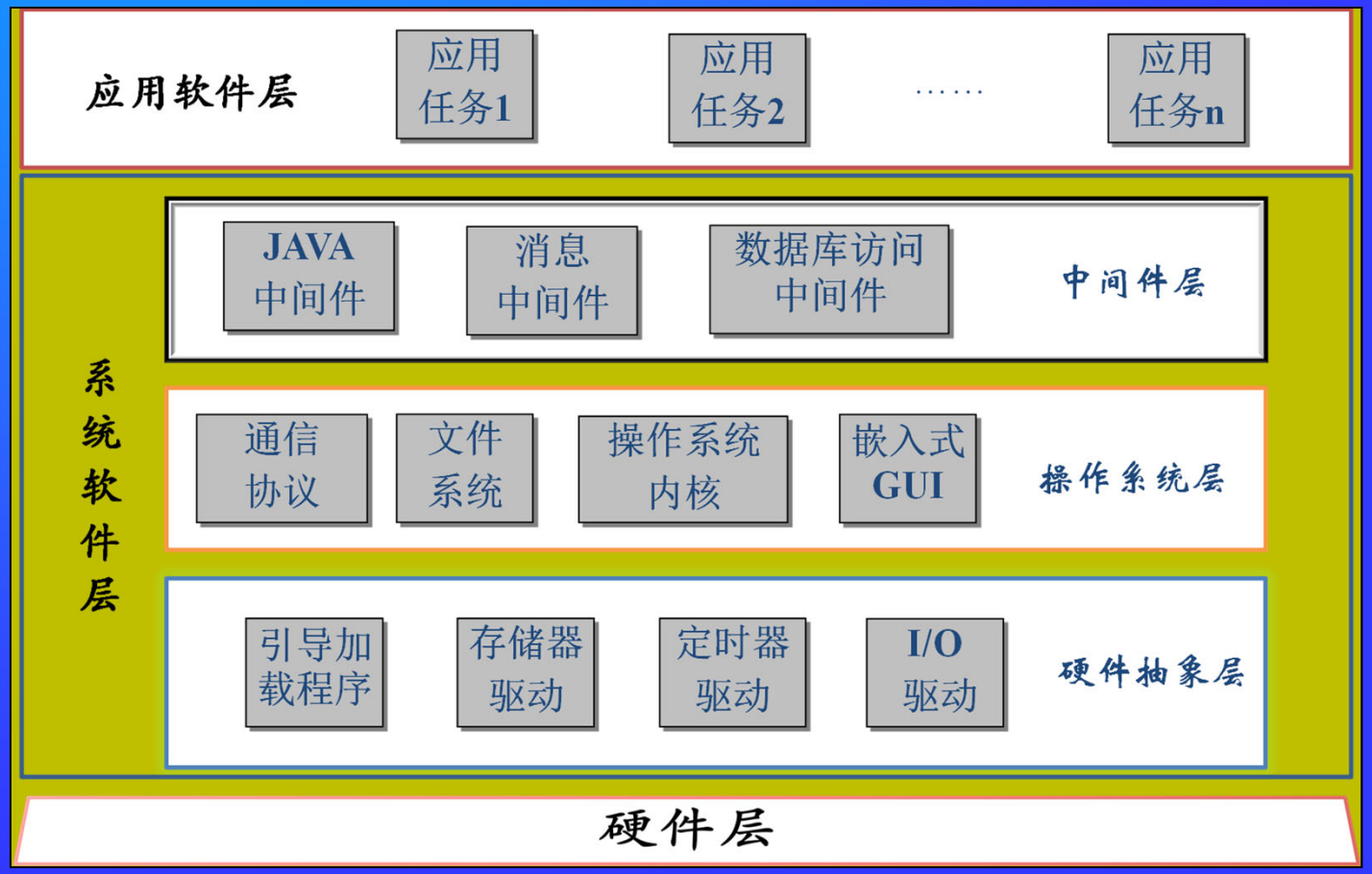
嵌入式系统软件的程序结构

- 简单监控式的嵌入式操作系统结构
  - 引入操作系统增加了系统的开销
  - 允许多个顺序执行的任务在一个CPU上并行运行，将复杂的系统分解成相对独立的多个任务
    - 降低了用户开发嵌入式软件的复杂度
    - 有效保证系统的实时性和可维护性等

嵌入式系统软件的程序结构

高性能通用嵌入式操作系统结构

高端嵌入式系统的软件层次结构实例



嵌入式系统软件的程序结构

- 简单循环轮询（Round-Robin）结构
- 带中断的循环轮询（Round-Robin）结构
  - 中断驱动结构/前后台结构
- 简单监控式的嵌入式操作系统结构
- 高性能通用嵌入式操作系统结构

- 选择嵌入式系统软件结构的基本原则：
  - 选择可以满足响应时间需求的最简单的结构

硬件抽象层（HAL）与板级支持包（BSP）

- 硬件抽象层（Hardware Abstract Layer，HAL）
  - 为了便于嵌入式操作系统在不同体系结构和硬件平台之间的移植，许多嵌入式环境要求在操作系统与硬件之间设置独立的软件层次
  - 所有直接依赖于硬件的软件，包括引导程序、硬件配置程序和硬件访问代码等
  - 将硬件抽象化，将系统上层软件与底层硬件分离开来，向操作系统上层提供统一接口，屏蔽硬件平台的细节和差异
  - 通过硬件抽象层应用编程接口（HAL API）向操作系统以及应用程序提供对硬件进行抽象后的服务
  - 将控制所有硬件电路的软件代码封装起来，使上层软件开发人员无需关心底层硬件的具体情况

硬件抽象层（HAL）与板级支持包（BSP）

- 常见的HAL规范大多是由操作系统厂商提出的
- 大多数操作系统为了实现硬件平台之间的可移植性，都会定义自己的硬件无关代码与硬件相关代码之间的接口规范
  - 硬件无关的操作系统代码在不同硬件平台上是相同的，由操作系统厂商提供
  - 依赖于硬件的代码通常要由用户或硬件平台制造商单独编写，这部分代码则称为**板级支持包**（Board Support Package，BSP）

硬件抽象层（HAL）与板级支持包（BSP）

- 依赖于硬件的代码通常要由用户或硬件平台制造商单独编写，这部分代码则称为**板级支持包**（Board Support Package，BSP）
- 板级支持包：
  - 屏蔽了其所支持的嵌入式操作系统与底层硬件平台之间的相关性
  - 使嵌入式操作系统能够通用于BSP所支持的硬件平台
  - 实现嵌入式操作系统的可移植性和跨平台性
  - 实现嵌入式操作系统的通用性和复用性

硬件抽象层（HAL）与板级支持包（BSP）

- 硬件抽象层：嵌入式系统软件的一个抽象层次
- 板级支持包：硬件抽象层在特定操作系统环境下的一种实现
- 在概念上：嵌入式系统软件中位于硬件和操作系统层之间的硬件相关软件层次统称为硬件抽象层
- 在实现上：依赖于特定操作系统的硬件抽象层实现称为板级支持包，少数不依赖特定操作系统的硬件相关软件的实现仍称为硬件抽象层
- 不依赖特定操作系统的HAL的数量很少，有些文献并不区分HAL和BSP

硬件抽象层（HAL）与板级支持包（BSP）

板级支持包的特点

- **硬件相关性：**
  - 与硬件的密切相关
  - 需要为操作系统提供操作和控制具体硬件的方法
- **操作系统相关性：**
  - 在同一硬件平台上支持不同嵌入式操作系统的BSP的组织结构、向上层提供的功能以及服务接口定义都不相同
  - 一种嵌入式操作系统的BSP不可能用于其他嵌入式操作系统

板级支持包的功能

- BSP在系统上电复位之后负责系统初始软硬件环境的建立
  - 系统上电时的硬件初始化：芯片级初始化和板级初始化
    - 对嵌入式处理器本身的初始化
      - 设置嵌入式处理器的工作模式、总线接口方式、系统内存配置、中断向量表配置等
    - 对系统其他硬件设备的初始化
    - 在系统上电复位时，一般并不需要对系统中的所有硬件进行初始化，只需对操作系统运行所必需的硬件进行初始化操作

板级支持包的功能

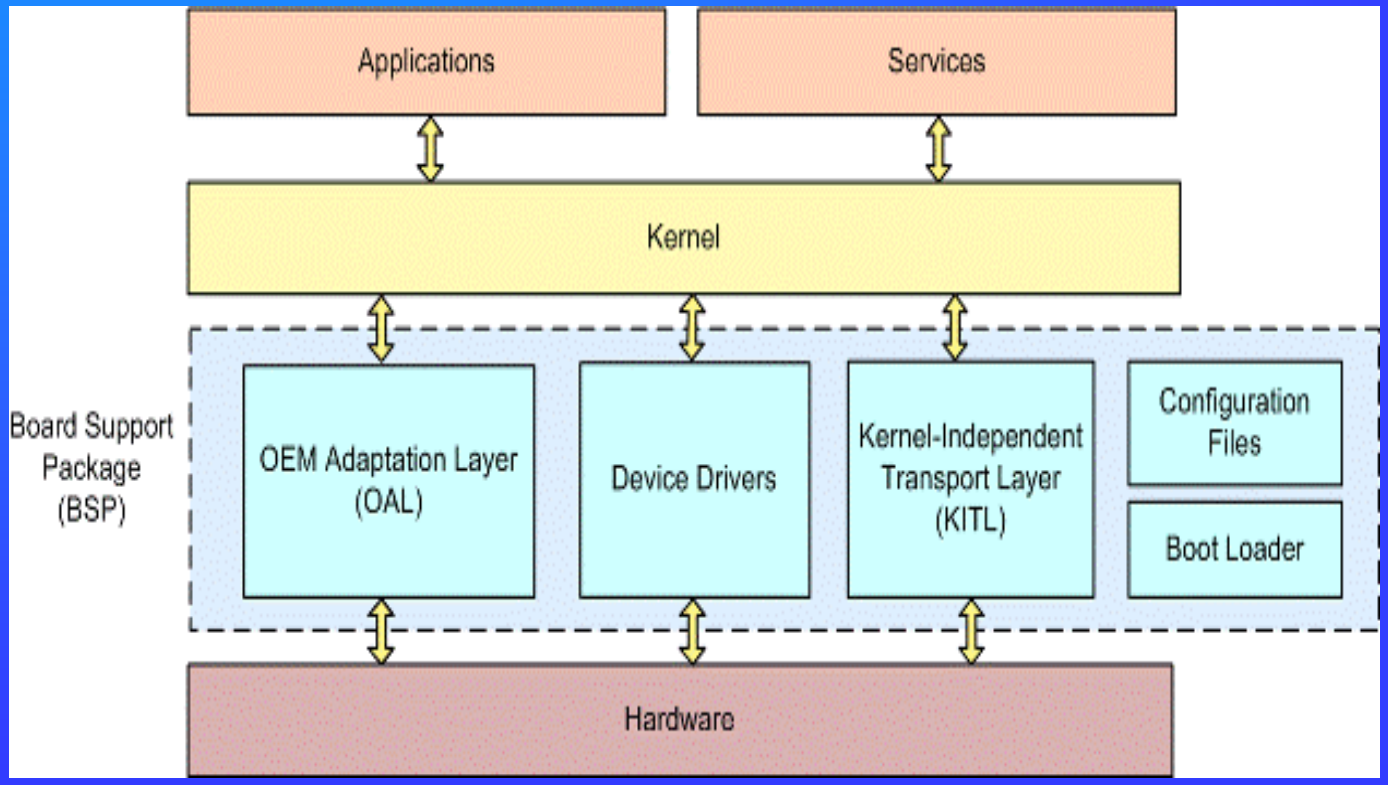
- BSP在系统上电复位之后负责系统初始软硬件环境的建立
  - 系统上电时的硬件初始化
  - 为操作系统提供硬件相关的驱动程序支持
    - BSP中包含硬件相关的设备驱动程序，为操作系统和/或应用程序访问硬件提供支持，对系统硬件进行管理，并实现数据输入输出操作
    - 除了与引导和加载操作系统所必需的硬件环境相关的设备驱动程序外，BSP包含的设备驱动程序通常不直接由BSP使用，而是在系统初始化过程中由BSP将其与操作系统中的通用设备驱动程序关联起来，并在随后操作系统运行之后由操作系统和/或应用程序调用，实现对硬件设备的操作

板级支持包的功能

- BSP在系统上电复位之后负责系统初始软硬件环境的建立
  - 系统上电时的硬件初始化
  - 为操作系统提供硬件相关的驱动程序支持
  - 集成操作系统加载模块
    - 进行系统级的初始化，包括设置软件的基本数据结构和参数、操作系统自身工作环境的初始化、定制操作系统的功能等
    - 加载操作系统，将系统的控制权转交给操作系统
    - 操作系统继续完成余下的初始化操作，例如加载和初始化与硬件无关的设备驱动程序，建立系统内存区，加载并初始化其他系统软件模块（如网络系统、文件系统）等

Windows Embedded Compact 7 的软件结构

- WEC7：微软公司WinCE的后续产品
- 专门为企业网络边缘的小型设备设计的安全、组件化的实时操作系统



引导加载程序（bootloader）

- 嵌入式系统上电复位后首先运行引导加载程序的代码，负责系统上电自检、硬件初始化、建立存储空间映射、配制系统参数、建立上层软件的运行环境，并加载和启动操作系统
  - 必备基本功能：引导（boot）和加载（load）操作系统等
  - 复杂的bootloader程序：简单的用户命令交互、设置操作系统启动参数、flash ROM编程下载、读写内存、系统自检、硬件调试等监控程序（monitor）功能

引导加载程序（bootloader）

- Bootloader依赖于具体硬件结构
  - 有些种类的bootloader能支持多种体系结构的处理器，但每种体系结构有其自身的版本
- bootROM：用来存储bootloader程序的非易失只读存储器芯片
  - 在多数嵌入式系统中，bootROM芯片内不仅存储bootloader程序，还存储操作系统映像、应用程序代码和用户配置数据等非易失信息

嵌入式操作系统的加载方式

- **加载方式一：在ROM中直接运行操作系统代码**
  - 将操作系统和应用程序的映像存放在非易失的ROM存储器中（通常是FLASH ROM）
  - bootloader代码完成必要的硬件初始化等操作后，跳转到ROM存储器中操作系统入口函数处继续执行
- **加载方式二：在RAM中运行操作系统代码**
  - bootloader代码在完成必要的硬件初始化等操作后，把存放在ROM存储器中特定位置的操作系统和应用程序映像拷贝到RAM存储器中，然后跳转到RAM存储器中操作系统入口函数处继续执行

嵌入式操作系统的加载方式

- **加载方式三：从外存储器加载操作系统代码运行**
  - 操作系统映象和应用程序映象存放在基于外存储器的文件系统之上
  - bootloader代码中必须要有访问外存储器所需要的驱动程序以及文件系统
  - 嵌入式处理器复位后，执行bootrom中存放的bootloader代码，完成必要的硬件初始化等操作后，通过外存驱动程序读取外部存储器，把操作系统和应用程序的映象复制到RAM中，然后跳转到操作系统入口运行
- **加载方式四：从通信端口加载操作系统代码运行**
  - 从串口、以太网接口等通信端口加载操作系统和应用程序
  - 需要在bootloader中增加通信端口驱动程序，往往还要有相应的通信协议软件支持通信端口上的信息交互过程

引导加载程序的操作模式

- 大多数bootloader包含两种不同的操作模式：
  - 启动加载（bootloading）模式：嵌入式系统正常工作时的独立启动方式，bootloader从非易失存储介质中引导和运行操作系统代码
  - 下载模式（downloading）：在调试或系统更新阶段使用的系统启动方式。bootloader通过通信端口从调试主机上下载操作系统内核映像，暂存到RAM中，然后将其烧录到flash ROM等固态存储器中
- 同时支持这两种工作模式：
  - 在进入启动加载模式之前延迟数秒钟时间并通过终端界面给出提示，如果用户按任意键则切换到下载模式，否则继续进行正常的启动加载

引导加载程序的主要任务与执行流程

- 大多数从固态存储设备上启动的bootloader程序采用两阶段启动过程
- 第一阶段（stage I）：
  - 让嵌入式系统能正常运行起来，并为第二阶段以及随后操作系统内核的运行准备好基本的硬件环境
  - 与处理器体系结构相关的硬件初始化和板级初始化等操作通常都在第一阶段完成
  - 代码通常用汇编语言实现
    - 提高代码的时空效率
    - 更方便地实现对硬件操作
  - 为高级语言程序的运行建立环境
  - 程序跳转到第二阶段的C程序入口点继续执行

引导加载程序的主要任务与执行流程

- 大多数从固态存储设备上启动的bootloader程序采用两阶段启动过程
- 第二阶段（stage 2）：
  - 代码通常用C语言实现，便于实现更复杂的功能
  - 实现操作系统加载，通常包括：
    - 建立系统配置信息交互通道和映像下载通道
    - 校验内核镜像
    - 解压缩内核镜像
    - 复制内核镜像到RAM
    - 为内核执行提供合适的上下文环境

嵌入式系统的设备驱动程序

- **设备驱动程序（device driver）的概念和功能**
  - 直接与硬件相互作用并控制硬件的软件
  - 从驱动程序调用者的角度看，驱动程序是对I/O操作的抽象，管理高层软件对硬件的访问
  - 应用软件工程师调用驱动程序实现数据包收发时不必关心硬件设备的中断、I/O寄存器、输入输出控制方式等硬件细节，从而实现对硬件的透明访问
  - 驱动程序的基本特性
    - 只能被高层软件调用而无法自行运行
    - 对上层屏蔽硬件细节，对下层直接操作硬件

嵌入式系统的设备驱动程序

- **设备驱动程序（device driver）的概念和功能**
  - 通常以若干函数的集合的形式出现，这些函数封装了对硬件的操作，例如设备初始化和释放、设备的读/写（输入/输出）、设备的控制、设备状态采集、设备错误处理等
  - 设备驱动与硬件直接打交道，按照硬件的具体工作方式读写设备接口内的寄存器，完成设备轮询、中断处理、DMA操作等；同时向高层软件提供一致和熟悉的接口
- **一般而言，完整的驱动程序通常包含硬件初始化、硬件使能、硬件工作模式和参数设置等功能，并提供驱动特定硬件设备工作的一组函数**

设备驱动程序的实现方式

- **设备驱动程序的实现方式依赖于系统所采用的嵌入式操作系统的驱动程序架构**
  - **不使用操作系统的嵌入式系统：**
    - 驱动程序和高层软件之间通常并没有严格的界限
    - 软件工程师可以根据不同类型的硬件设备的特点自行定义硬件操作接口
  - **小型的嵌入式操作系统：**
    - 只包括操作系统内核，不包括设备驱动程序
    - 在操作系统中编写设备驱动程序也没有定义统一的驱动框架
    - 驱动程序开发完全由软件工程师自行掌握，实现设备驱动功能的代码注注也与其他用户代码混合在一起

设备驱动程序的实现方式

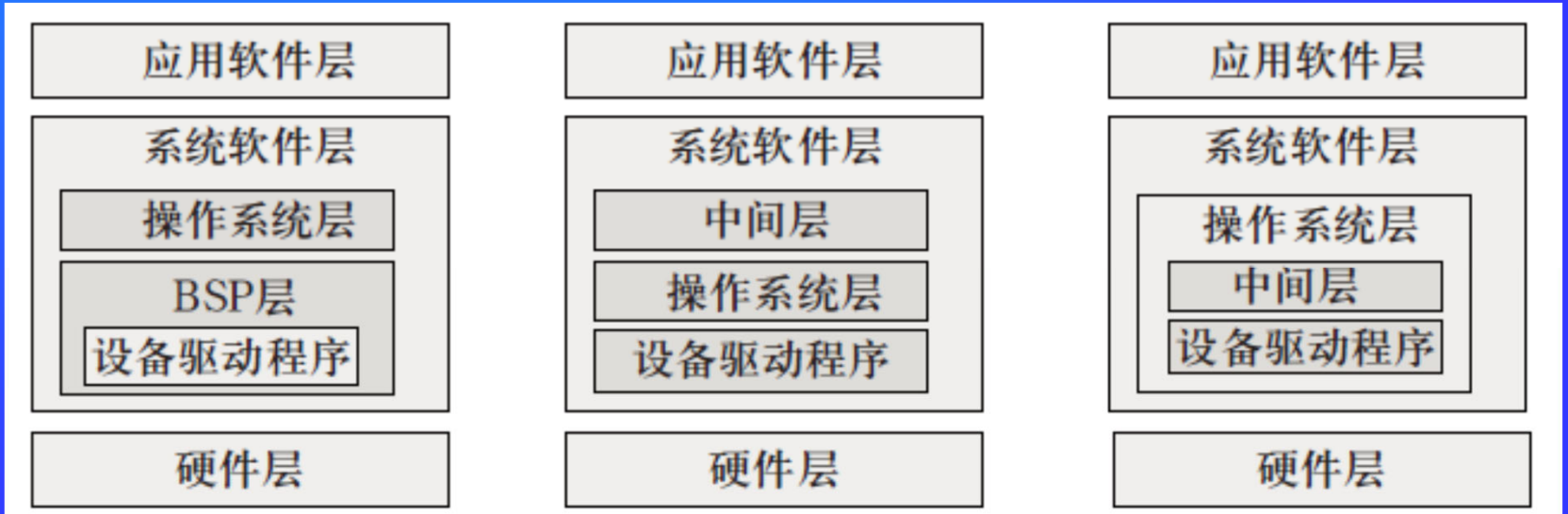
- 设备驱动程序的实现方式依赖于系统所采用的嵌入式操作系统的驱动程序架构
  - 不使用操作系统的嵌入式系统
  - 小型的嵌入式操作系统
  - 复杂的大型嵌入式操作系统：驱动程序架构严格遵从操作系统的定义
    - 注注自带一些标准设备的驱动程序
    - 定义了统一的设备驱动程序架构
    - 驱动工程师必须按照相应的架构开发自己的设备驱动程序，才能与操作系统内核相互协作
    - 某些操作系统（例如VxWorks）也允许用户代码绕过操作系统直接访问硬件
      - 从功能上与驱动程序的概念相符
      - 从内核的角度看属于应用代码

设备驱动程序的实现方式

- 设备驱动程序的实现方式依赖于系统所采用的嵌入式操作系统的驱动程序架构
  - 不使用操作系统的嵌入式系统
  - 小型的嵌入式操作系统
  - 复杂的大型嵌入式操作系统：驱动程序架构严格遵从操作系统的定义
- 一些操作系统把设备进一步抽象成设备文件，以便高层软件更方便地访问硬件设备
  - 硬件设备只是一个设备文件
  - 应用程序可以像操作普通文件一样对硬件设备进行操作

驱动程序所处的层次位置

- 不同的嵌入式操作系统中驱动程序所处的层次位置并不相同
  - 设备驱动层是板级支持包的一部分
  - 设备驱动程序在操作系统层之下单独构成一层
  - 设备驱动程序与操作系统内核都处在操作系统层中



实时系统与实时操作系统

- **实时系统（Real-Time System）：必须在有限和确定的时间内对外部输入事件做出响应的信息处理系统**
  - **IEEE计算机协会实时系统技术委员会（IEEE-CS TCRTS）给出的实时系统定义：**
    - 正确性不仅取决于计算的结果，而且取决于产生结果的时间的计算系统
    - 实时系统对外界的响应是否正确不仅取决于功能正确性，而且取决于对事件处理的时间正确性

实时系统与实时操作系统

- **实时系统（Real-Time System）：必须在有限和确定的时间内对外部输入事件做出响应的信息处理系统**
  - **评判实时系统的关键：**
    - 不是系统对外部事件的处理速度，而是处理事件的时间的可预见性和确定性
    - 无论系统面对最小负载还是最大负载，实时系统都必须确定性地保证满足时间要求
      - **非实时的通用计算机系统：追求系统的平均响应时间短和用户使用的方便性**
      - **实时系统：主要考虑的是在最坏情况下的系统行为的可预见性是否有保证**

实时系统与实时操作系统

- 时限（deadline，最后期限）：
  - 相对时限：请求发出后，该系统作出响应所需要的时间
  - 绝对时限：不管任务何时开始，一个任务必须完成的截止时间

实时系统中运行的任务

- **硬实时任务**（hard real-time task，**有时也称强实时任务**）
  - 必须在给定的时限内响应，错过最后期限会导致任务失败或引起致命的错误
  - 实例：工业控制和军工系统中的任务
- **软实时任务**（soft real-time task）
  - 仍然要求系统的响应越快越好，但是偶尔超出时限并不会造成任务失败或导致任务出现致命错误，而是造成任务的服务质量下降，超时的响应仍然有一定的意义
  - 实例：一些消费类电子产品和数据采集系统中的任务

实时系统中运行的任务

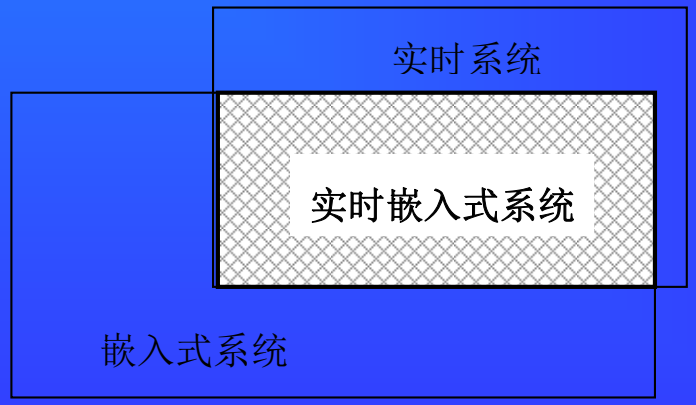
- **硬实时任务**（hard real-time task，**有时也称强实时任务**）
- **软实时任务**（soft real-time task）
- **准实时任务**（firm real-time task，**有时也称弱实时任务**）
  - **允许偶尔错过最后期限，但若超过时限才响应，处理任务所进行的操作或者计算结果没有任何意义**
- **一个实时系统所要处理的事件通常不止一个，而多个任务的实时性要求各有不同，因此单一的实时系统中往往包含上述几种类型的任务**

实时系统中运行的任务

- 硬实时任务（hard real-time task，有时也称强实时任务）
- 软实时任务（soft real-time task）
- 准实时任务（firm real-time task，有时也称弱实时任务）
- 一个实时系统所要处理的事件通常不止一个，而多个任务的实时性要求各有不同，因此单一的实时系统中往往包含上述几种类型的任务
- 实时性要求高的系统既要有足够强大的硬件平台支撑，也要有合理的任务优先级安排，以满足实时处理的确定性要求和性能要求

实时系统与嵌入式系统

- 实时系统与嵌入式系统的关系
  - 嵌入式系统并不都是实时系统
  - 实时系统也不都是嵌入式系统
  - 实时嵌入式系统是嵌入式系统与实时系统的交集



实时操作系统（RTOS）

- **能够满足实时系统需求的操作系统就是实时操作系统（RTOS，Real-Time Operating System）**
  - **不以在给定时间内完成更多的任务为目标，而是以响应外部事件尽可能快、且响应时间具有确定性为目标的操作系统**
  - **最关键特性：任务处理时间的一致性**
    - 如果一个实时操作系统能够完全满足确定性的时限要求，则称为**硬实时操作系统**，否则称为**软实时操作系统**
  - **实时操作系统完成连续任务的时间长度的偏差称为抖动（jitter）**
    - 硬实时操作系统的抖动比软实时操作系统小

实时操作系统的时间确定性

- 对于支持硬实时任务的操作系统而言，系统调用与服务的执行时间都应具有可确定性
  - 系统服务的执行时间不依赖于应用程序任务的多少，而且系统完成某个确定任务的时间是可预测的
  - 整个系统必须满足一定的峰值负荷要求，即在重负载甚至超负荷的情况下，某些关键任务一定要得到服务
  - 操作系统可以确定性地满足请求的程度取决于
    - 响应中断的速度
    - 系统是否具有足够的处理能力在要求的时间内处理所有的请求

实时操作系统的时间确定性

- **实时操作系统依赖相应的机制保证对实时任务的响应的时间确定性**
  - **事件驱动**
    - 能对来自外界的作用和信号在限定的时间范围内作出响应
    - 外部事件：
      - **周期性的同步事件，系统可以预见下一次同类事件发生的时刻**
      - **非周期性的异步事件，事件发生的时间不可预测**
  - **多级中断嵌套处理**
    - 需要建立多级中断嵌套处理机制，以确保对紧迫程度较高的实时事件进行及时响应和处理

实时操作系统的时间确定性

- **实时操作系统依赖相应的机制保证对实时任务的响应的时间确定性**
  - **细粒度任务优先级控制**
    - 当前最高优先级的任务一旦就绪，总能立即得到处理器的使用权
    - 高优先级任务确定先执行，且在任务执行中不会被低于其优先级的其他任务打断或抢占资源
    - 同等优先级任务则建立任务队列，按照队列先入先出规则获得资源或执行
  - **实时抢占式调度**
    - 实时调度机制的目标：
      - **在正常情况下尽可能满足所有任务的时限**
      - **在峰值负载条件下保证强实时任务的时限**
    - 选择调度方式简单、反应速度快的实时调度算法
    - 建立更多“安全切换”时间点，保证及时调度实时任务

典型嵌入式操作系统

- 与通用计算机平台不同，没有一款产品能够占据嵌入式操作系统绝对统治地位
- 一些应用面相对较广、产品特性突出的嵌入式操作系统有相当大的知名度

风河系统公司的VxWorks

- 美国风河系统公司（WRS，Wind River System，2009年-2018年为英特尔全资子公司）于1983年专门针对嵌入式环境开发的商用嵌入式实时操作系统
- 具有高性能的内核、友好的用户开发环境
- 良好的可靠性和卓越的实时性
- 广泛应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中
- 提供完整的基于开发主机的集成开发环境Tornado
  - 编辑、编译、连接、调试和存储实时代码
  - VxWorks 6.0以上版本：Workbench开发平台
    - 继承Tornado集成开发平台的一贯优势
    - 采用Eclipse软件框架结构
    - 广泛适用多任务、多目标、多模式、多操作系统、多处理器、多连接形式、多主机环境的开发要求

风河系统公司的VxWorks

- 支持Power PC、68K、CPU32、i960、x86、MIPS等等多种嵌入式处理机体系结构
- 采用微内核结构
  - 内核仅提供多任务环境、任务间通信和同步功能
    - 有效保证较短的任务间切换时间和较小的中断延迟
- 较好的可剪裁能力
- 应用程序的动态连接和动态下载
- 较好的兼容性
- 高可靠性、高可用性和高安全性

QNX软件系统公司的QNX

- 由加拿大QNX软件系统公司（2010年被黑莓公司收购）开发的实时操作系统
- 商业分布式、可扩充的实时操作系统
- 应用范围极广：
  - 汽车领域、网络通信、核电站和无人驾驶坦克的控制系统
  - RIM公司的BlackBerry PlayBook平板电脑
- 类Unix实时操作系统
  - 多数传统Unix程序在微量修改后即可在QNX上编译与运行
  - 提供具有UNIX特色的编译器、调试器、X Window和TCP/IP协议栈

QNX软件系统公司的QNX

- 微内核操作系统
  - 小巧且运行速度极快的微内核以及一些可选的配合进程
    - 内核仅提供4种服务：进程调度、进程间通信、底层网络通信和中断处理
    - 所有其它服务通过协作用户进程实现
      - 可以灵活实现系统的可伸缩和动态配置
  - QNX同时支持进程和线程两种任务实现方式
    - 保证应用系统稳定、可靠、强壮

微软公司的嵌入式Windows操作系统家族

- 1996 年发布Windows CE 1.0
  - 精简的Windows 95操作系统的嵌入式版本
  - 设计目标：使嵌入式开发人员能够将“当今个人电脑复杂的软件环境扩展到嵌入式世界”
- 2008年，Windows CE系列更名为Windows Embedded Compact

微软公司的嵌入式Windows操作系统家族

- 微软的嵌入式操作系统家族产品
  - 嵌入式操作系统（EOS）
    - Windows Embedded Compact（WEC）系列
      - 专为需要弹性硬件和硬实时支持的小型设备而设计的平台
      - 使用完全重新设计的核心，与通用平台上的Windows核心不同，却拥有与桌面Windows家族一致的程序开发界面和基于Win32的应用程序接口
        - » 绝大多数的桌面应用软件只需简单的修改和移植就可以在WEC平台上继续使用
        - » 曾经在消费电子领域占据很大的市场份额

微软公司的嵌入式Windows操作系统家族

- 微软的嵌入式操作系统家族产品
  - 嵌入式操作系统（EOS）
    - Windows Embedded Compact（WEC）系列
    - Windows Embedded系列
      - 针对既非小型设备、又非资源受限设备的嵌入式环境
      - 运行该系列操作系统的硬件平台通常比WEC系列的平台具备更大的尺寸，且硬件资源更加丰富
      - 基于相同版本的Windows客户操作系统
        - Windows Embedded 8源于Windows 8
        - 包含了Windows 8的绝大多数功能
        - 增加了一些嵌入式环境所需的特性

微软公司的嵌入式Windows操作系统家族

- **微软的嵌入式操作系统家族产品**
  - **嵌入式操作系统（EOS）**
  - **设备平台（device platforms）**
    - 以两个EOS系列产品为基础，构建针对特定行业的设备平台
    - 在特定硬件平台上为特定应用预定制的Windows嵌入式操作系统实例
    - 设备制造商可以在这类平台上采用特殊组件和功能应用完成平台的客户定制
      - **加快开发过程，缩短上市时间**
      - **例：基于Windows Phone 8平台的Windows Embedded 8 Handheld（手持版）**
        - » 用于手持设备的设备平台
        - » 可以利用 Windows Phone 8 SDK开发应用

RTEMS操作系统

- RTEMS：实时多处理器系统
- GPL开源的无版税嵌入式实时操作系统
- 由美国军方主导研制
  - 最早用于美国的国防系统，目前由OAR公司负责维护
- 微内核抢占式实时系统
- 稳定且运行速度快
- 占用系统资源小
- 支持多处理器
- 提供完整的TCP/IP协议栈以及FTP、WebServer、NFS等服务
- 在性能上与VxWorks相比毫不逊色
- 在通信、航空航天、工业控制、军事等领域应用广泛

FreeRTOS

- 轻量级的开源迷你实时操作系统内核
- 完全免费且源码公开
- 可移植、可裁减、调度策略灵活
- 内核小巧，占用存储空间少，可以方便地移植到各种单片机上运行

作业

1. 按实时性分类，下列嵌入式操作系统中不属于硬实时操作系统的是（　　）。

A：RTEMS　　B：WinCE

C：μC/OS-II　　D：VxWorks

2. 下面的选项中不属于实时系统关键特性的是（　　）。

A：可配置性　　B：可预测性

C：时间约束性　　D：可靠性

3. 对实时操作系统而言，无助于提高实时响应的时间确定性的机制是（　　）。

A：软件固化存储　　B：事件驱动

C：多级中断嵌套处理　　D：实时抢占式调度

作业

4. 下列关于嵌入式系统的软件结构的描述中，不正确的是：

A. 简单的轮询结构只适用于系统的任务数量较少、任务处理简单，且实时性要求不高的场景。

B. 选择嵌入式系统软件结构的一个基本原则是：选择可以满足响应时间需求的最简单的结构。

C. 与简单的轮询结构相比，带中断的轮询结构由于中断的引入而使系统软件的复杂度明显降低。

D. 带中断的轮询结构也称为中断驱动结构或前后台结构。

5. 下列关于硬件抽象层和板级支持包的概念的陈述中，不正确的是：

A. 板级支持包的特点是与硬件和操作系统都关系紧密，既有硬件相关性，又有操作系统相关性。

B. 硬件抽象层是在操作系统层与硬件之间设置的独立的接口软件层，是所有直接依赖于硬件的软件。

C. 板级支持包中包含硬件相关的设备驱动程序。

D. 板级支持包主要完成内存加电自检、外设存在自检、初始化外围设备、加载和启动操作系统等功能。

作业

6. 说明硬件抽象层（HAL）与板级支持包（BSP）的含义
7. 嵌入式操作系统有几种加载方式？哪种加载方式加载速度最快？哪种加载方式操作系统运行速度最快？
8. 一个洗衣机的控制程序需要检查面板上的按键的状态，并在监测到水位异常、运行中开盖等事件时发出告警信息，此程序适合采用哪种软件结构？

本章结束