# 现 代 操 作 系 统

# Modern Operating Systems

宋虹 songhong@csu. edu. cn

中南大学计算机学院网络空间安全系

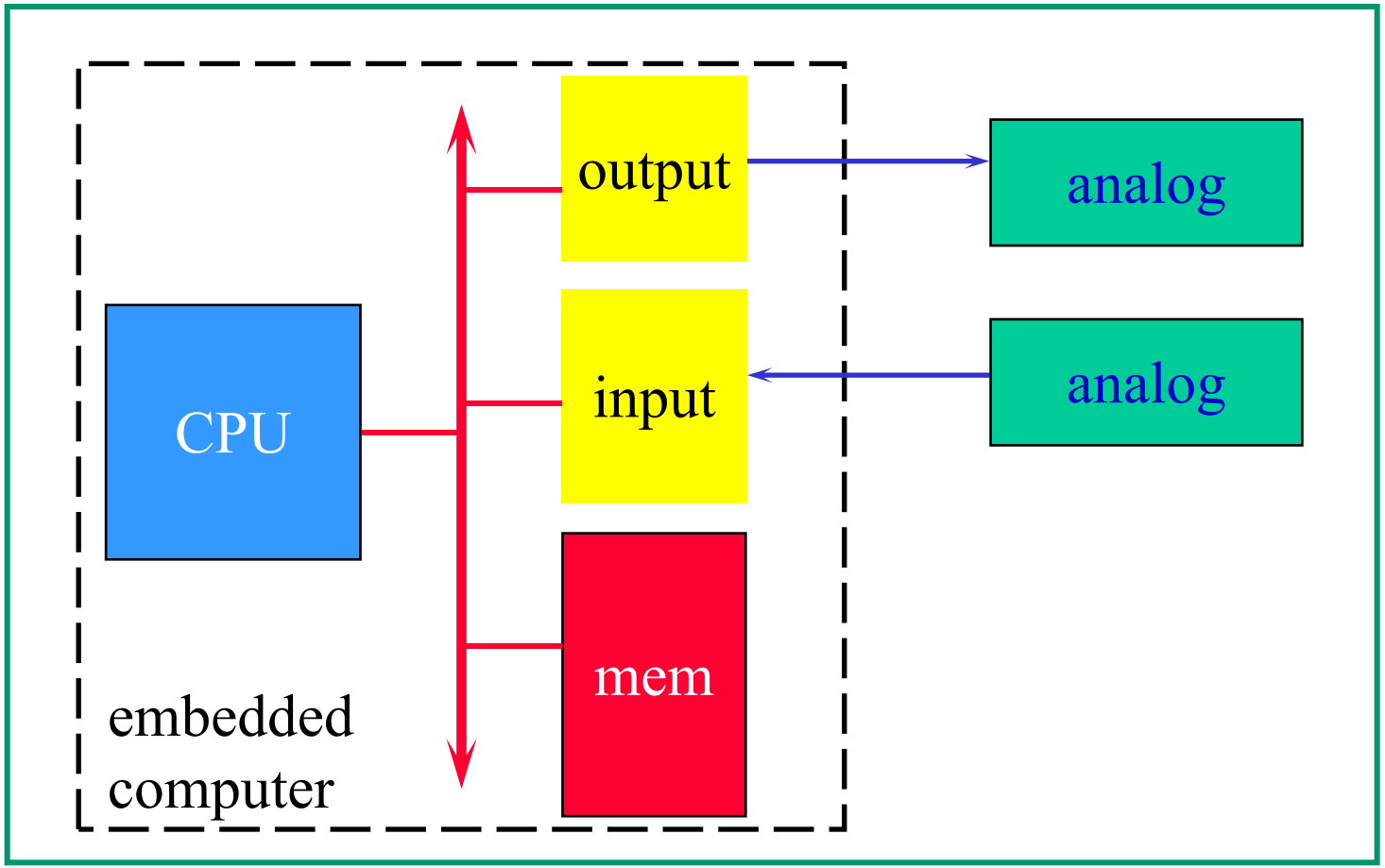
# Chapter 8 Embedded Operating Systems

- **8.1 Introduction to Embedded System**
- **8.2 Introduciton to  Embedded OS**
- **8.3 Task Management**
- **8.4 Embedded File System**

# 8.1 Introduction to Embedded System

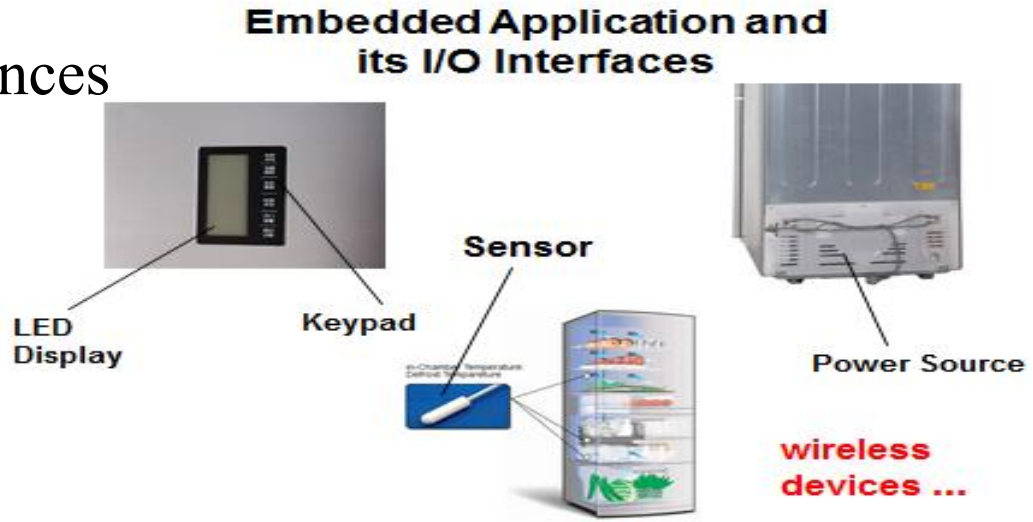
- **Definition**
  - An Embedded System is one that has a computer hardware along with one of its most important component- the 'software', which embeds into it.
  - Embedded Systems are the electronic systems that contain a microprocessor or a microcontroller, but we do not think of them as computers- the computer is hidden or embedded in the system.
  - Specialized computer system

# Embedding a computer



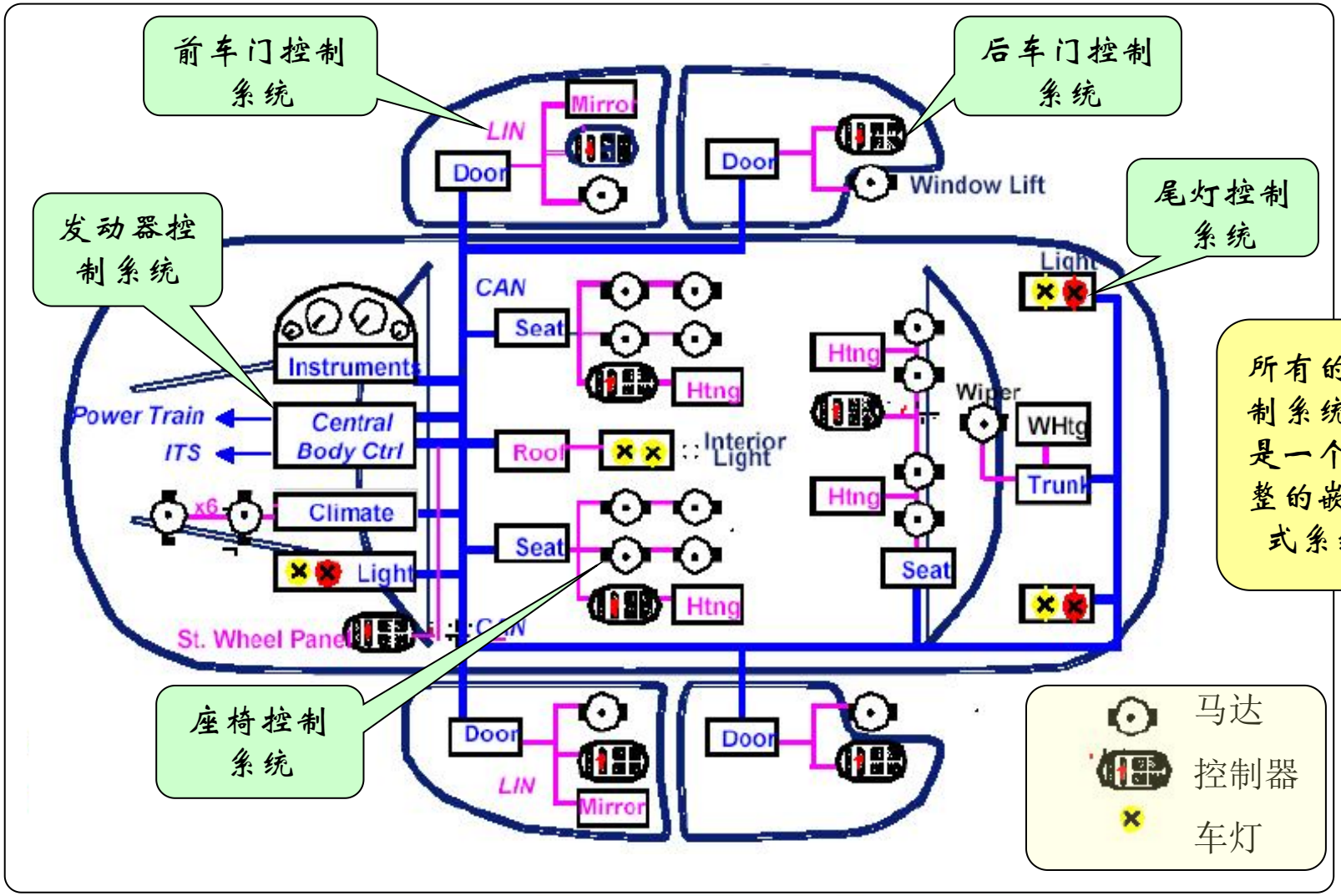
# 8.1 Introduction to Embedded System

- **Examples**
  - PDA: Personal digital assistant
  - Printer
  - Cell phone
  - Automobile: engine, brakes, etc.
  - Television
  - Household appliances

# Automobile Control System

# Automobile Control System



# 8.1 Introduction to Embedded System

- **Three classes**
  - Small scale system
    - single 8 or 16 bit microcontroller
    - little hardware and software complexities
    - C——development platform
    - limited power dissipation
    - ……
  - Median scale system
    - microcontroller——16 or 32 bit, DSP or RISCs
    - complex software design tools
      - C: source code engineering tool
      - RTOS, IDE: development platform
    - ……
  - Sophisticated system: enormous hardware and software complexities

# 8.1 Introduction to Embedded System

- **Hardware Elements**

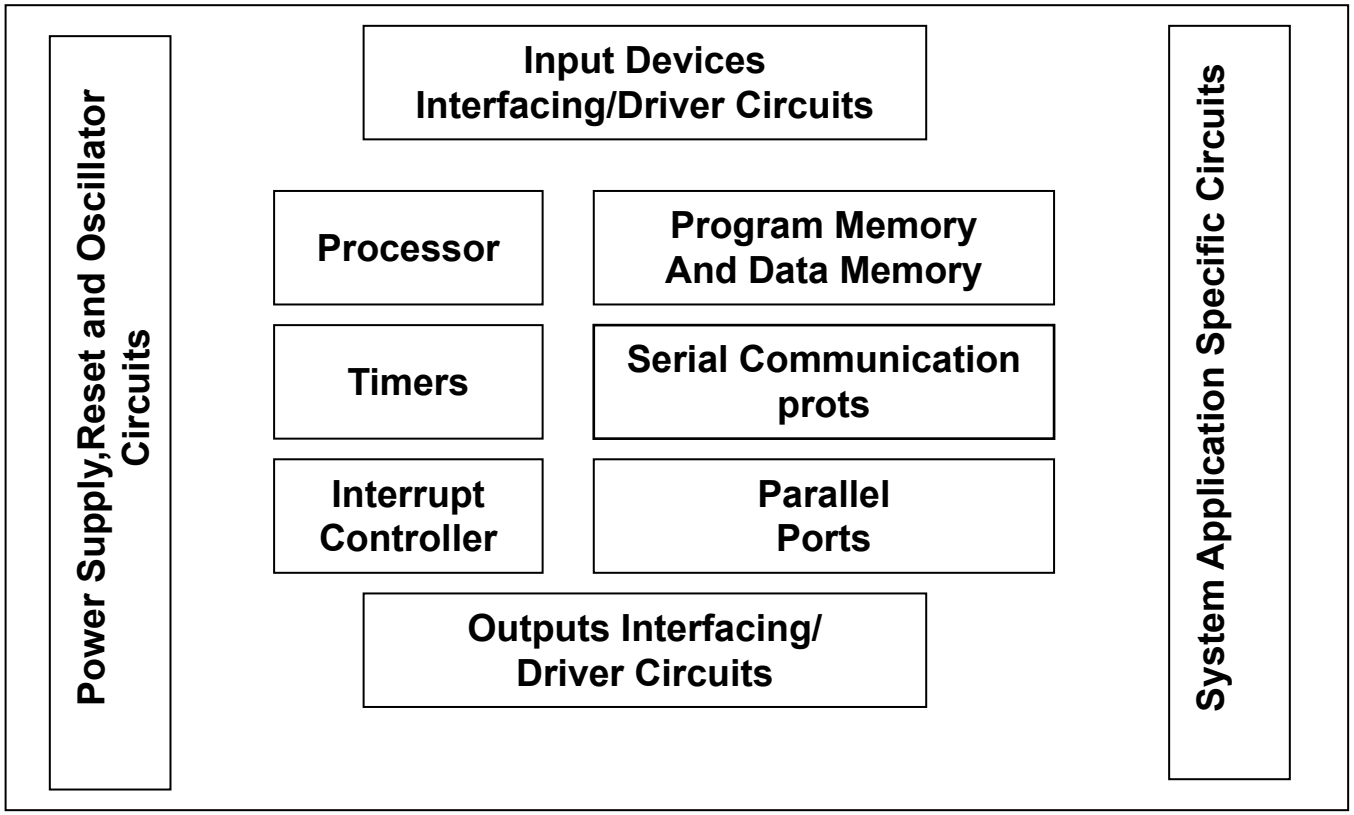



**Figure 8.1 The components of an embedded system hardware**

# 8.1 Introduction to Embedded System

- **Hardware Elements**
  - **Processor**
    - General purpose microprocessor
    - Application Specific System Processor (SSP)
    - Application Specific Instruction processor (ASIP)
    - Multiprocessor System using GPPs
  - **Basic Circuit Elements**
    - Power Source
    - Clock Oscillator Circuit, Clocking Units
    - Reset Circuit
    - Memory
    - Interrupts Handler
    - Linking Embedded System Hardware
  - **I/O Communication Unit**
  - **Memory**

# 8.1 Introduction to Embedded System

- **software Elements**
  - ROM Image
  - Assembly Language Codes
  - Application software
  - Application Software Development Tools
  - Device in a system
  - RTOS
  - Simulator

# 8.1 Introduction to Embedded System

**Application software**

- 'C' program has various layers:
  - processor commands
  - main function, task functions and library functions
  - interrupt service routines
  - kernel (scheduler)
- The compiler generates an object file
- Using linker and locator, the file for ROM image is created for the targeted hardware
- C++ and Java are other languages used for software coding.

# 8.1 Introduction to Embedded System

**Application software Development Tools**

- Source Code Engineering Tools
- tracks the switching from one task to another as a function of time, stores beats
- Trace Scope (traces changes in a parameter as a function of time)
- other Development Tools:
  - Editor
  - Interpreter
  - Compiler
  - Assembler and Cross
  - IDE

# 8.1 Introduction to Embedded System

## Devices in a System

- physical devices: keypad, LCD display, disk, parallel port, network-card
- **A device driver** is software for controlling, receiving and sending a byte or a stream of bytes from or to a device
  - Placing appropriate bits at the control register or word.
  - Calling an ISR on interrupt or on setting a status flag in the status register and run (drive) the ISR (also called Interrupt Handler Routine).
  - Resetting the status flag after interrupt service

# 8.1 Introduction to Embedded System

**Devices in a System**

- device management software provide codes
  - for detecting the presence of devices
  - for initializing these devices
  - for testing the devices that are present

# 8.1 Introduction to Embedded System

**Real Time Operating System (RTOS)**

- tasks for the system have real time constraints and deadlines for finishing the tasks
- Important RTOSs
  - μC/OS-II
  - Windows CE
  - VxWorks
  - Embedded Linux
  - Red Hat  e-COS (Embedded Configurable OS)

# 8.1 Introduction to Embedded System

**Simulator**

– To simulate the target processor and hardware elements on a host PC and to run and test the executable module

# 8.2 Introduction to Embedded OS

- **principal role** is the interaction with the physical world
- be usually written for special-purpose hardware or computer CPU chips differ from general-purpose CPUs
- be simple or be sophisticated
- Communications protocols: available as closed source.
- Open-source protocols:uIP, lwip, and others

# 8.2 Introduction to Embedded OS

- Features
  - compact——very specific
  - efficient at resource usage——limited resources
  - scalable
  - reliable
- difference between most RTOS & general OS
  - The applications be not loaded.
- instances of RTOS
  - Symbian OS
  - iOS
  - Android……

# 8.2 Introduction to Embedded OS

- more important factors in RTOS
  - runtime libraies
  - threads
  - CPU cycles
  - switched tasks
- performance metrics
  - guarantee of a soft or hard deadline
  - minimal interrupt latency
  - minimal thread switching latency
  - reliability: how predictably——Downtime
  - throughput

# 8.2 Introduction to Embedded OS

- components in RTOS
  - kernel
  - TCP/IP network system
  - File system
  - Graphics system
- Functions
  - Task management
  - time management
  - memory management
  - Communication, synchronization, and mutual exclusion mechanisms
  - interrupt management

# 8.3 Task Mangement

## 任务管理

- 任务管理是内核的核心，具有任务调度、创建任务、删除任务、挂起任务、解挂任务和设置任务优先级等功能。

- 什么是任务（Task）：每个操作系统都有一个最小的运行单位，系统为它分配资源，进行调度。在RTOS中，这个最小的运行单位就是任务。有些系统，最小的运行单位为进程（Process），有的则为线程（Thread）。

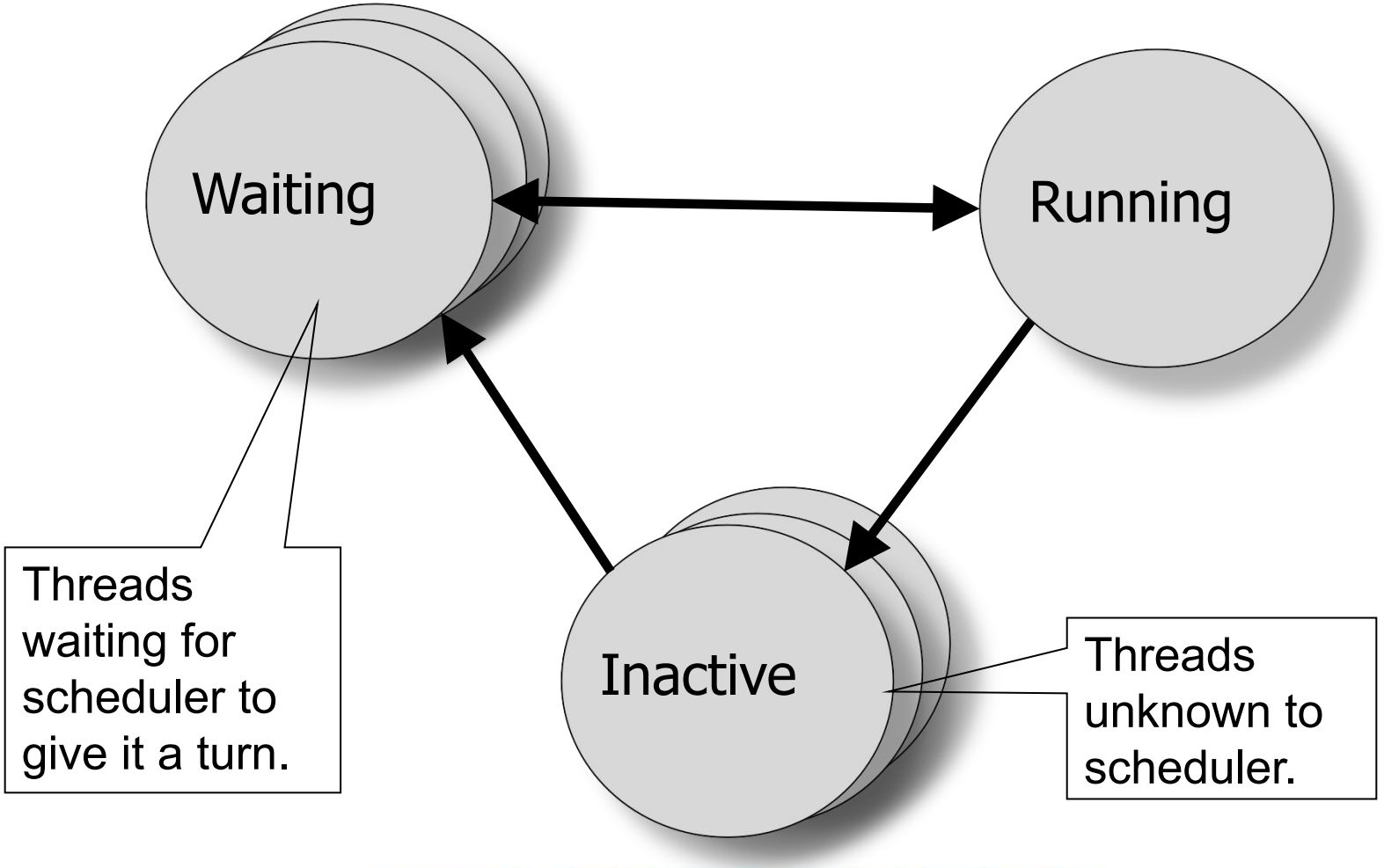
# 8.3 Task Mangement

- **进程与线程**
  - 对于通用计算机系统来说，进程代表程序的一次运行，而一个进程又是由一个以上的线程组成
  - The only worth-to-mention difference is that threads share the same address space, while processes don't.
- **任务控制块（TCB）**
  - TCB包括任务相关信息的数据结构和任务执行过程中所需要的所有信息
  - 任务管理是通过对TCB的操作来实现的
  - 每个任务都有唯一的任务控制块，TCB是任务在系统中存在的唯一标志

# 8.3 Task Mangement

- **任务状态**
  - 就绪态（Ready）：任务准备运行，但是还不能运行，因为一个更高的优先权的任务在执行
  - 阻塞态（Blocked）：请求一个还不能用的资源；等待某些事件发生；自身延迟一段时间
  - 运行态（Running）：任务是最高优先权的任务，正在运行
  - 状态转换：
- **调度器scheduler**——调度算法，函数形式
- **调度点**——调用调度器的具体位置
- **调度算法：**离线/在线调度算法——RMS和EDFS

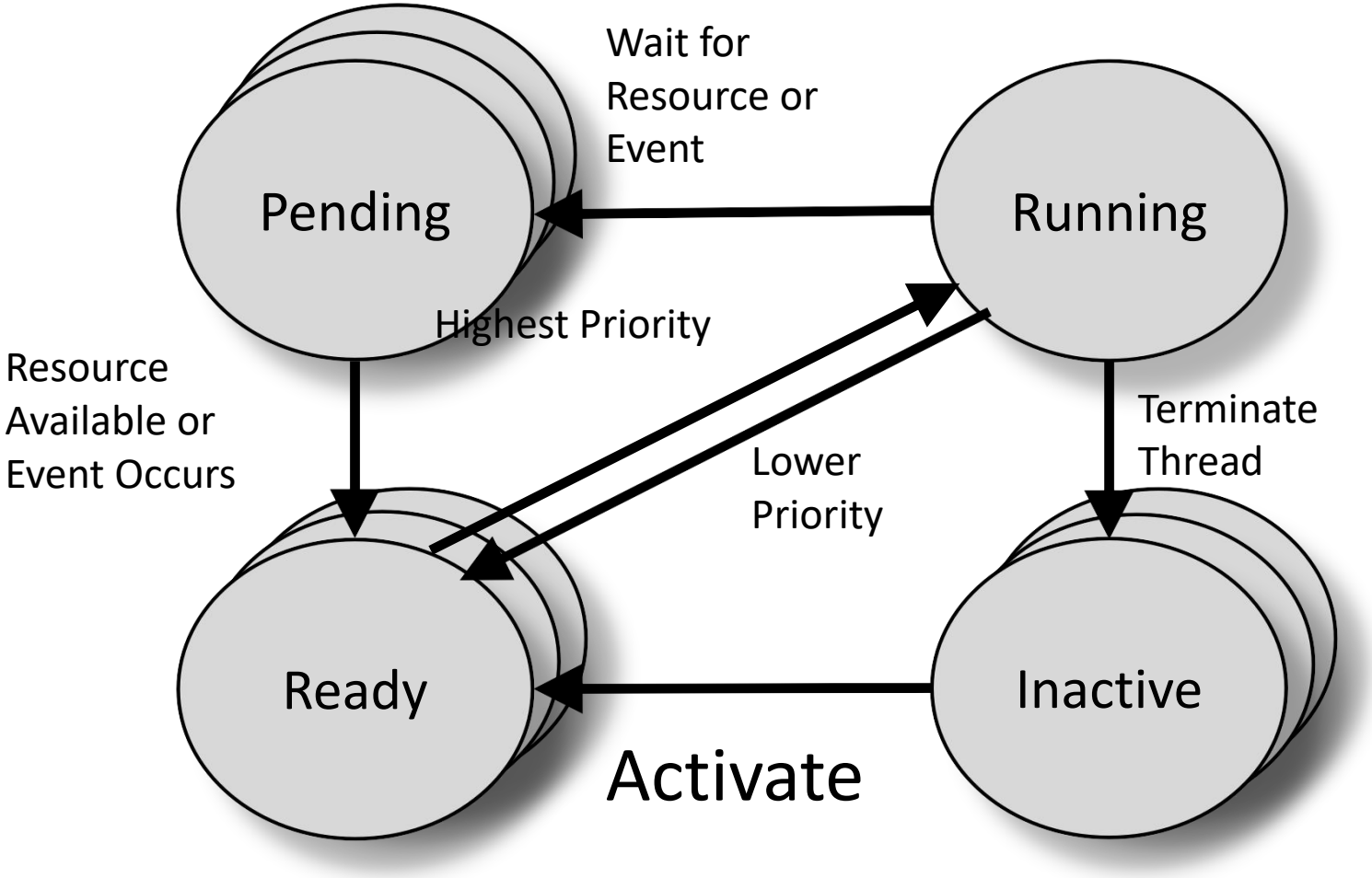
# 8.3 Task Mangement



# 8.3 Task Mangement

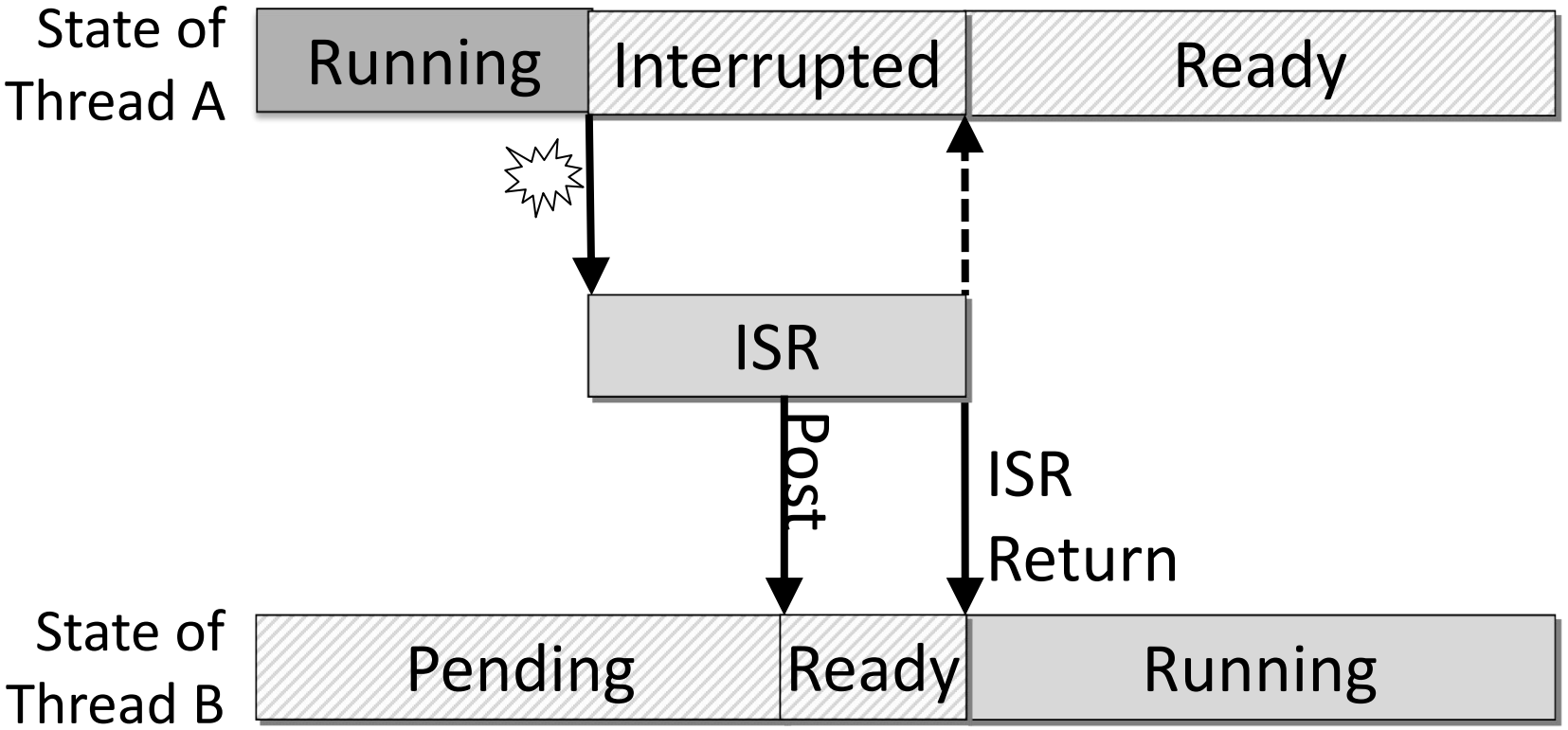
- **任务调度**
  - 通用计算机操作系统的调度原则是公平，调度的时机主要以时间片为主驱动
  - 嵌入式实时系统多采用基于优先级的抢占式调度策略
- **时间片（Time Slice、Time Slot)**
  - 简单的说，就是CPU分配给各个程序的时间，使各个程序从表面上看是同时进行的，而不会造成CPU资源浪费
  - 为什么采用时间片?
    - 宏观：同时打开多个应用程序，同时运行
    - 微观：只有一个CPU，一次只能处理程序要求的一部分，引入时间片，每个程序轮流执行

# 8.3 Task Mangement



# 8.3 Task Mangement

- **Preemptive Context Switch**



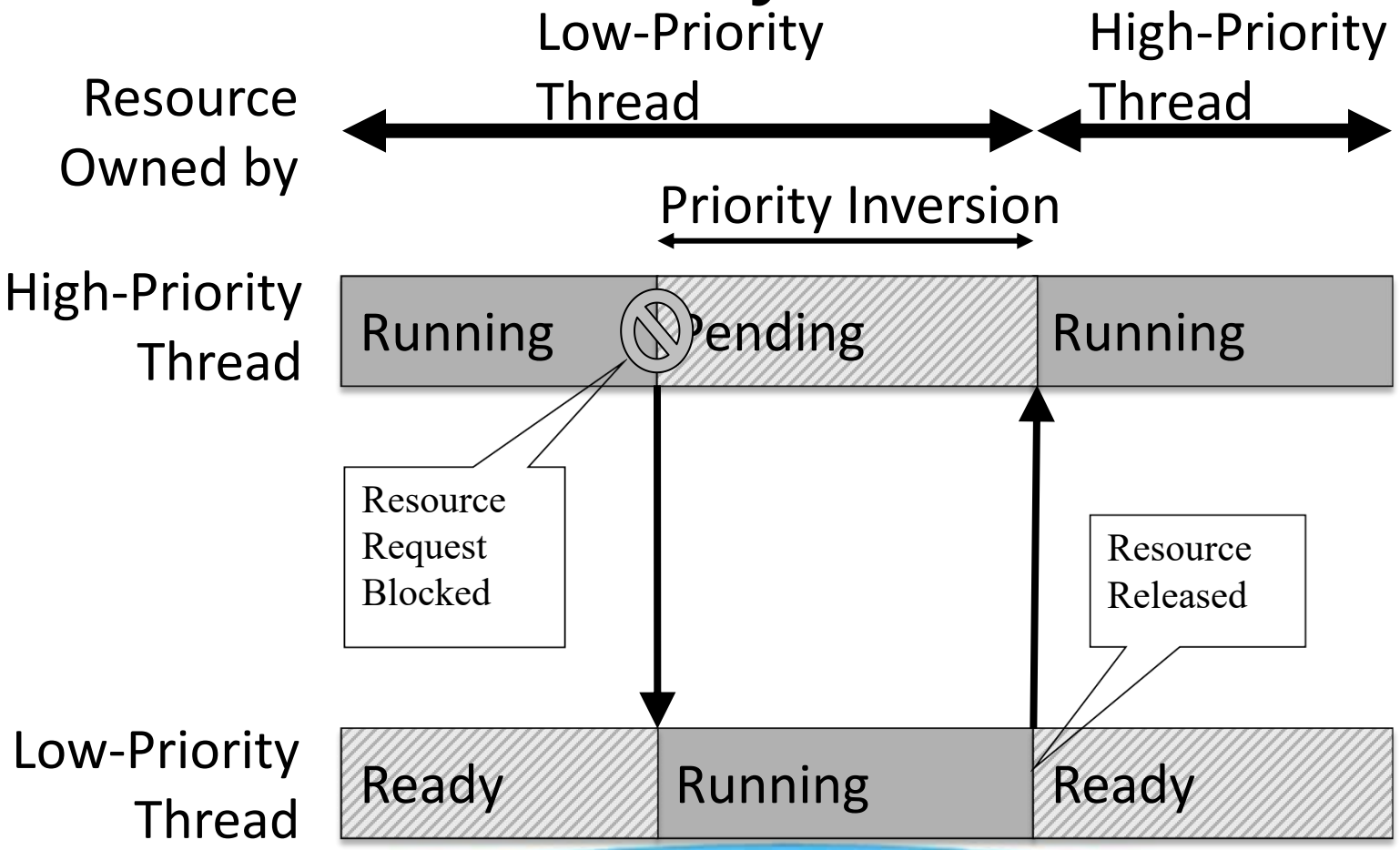
# 8.3 Task Mangement

## Problem——Priority Inversion

- 在实际开发中，由于任务间资源共享，信号量及中断的引入，往往会出现高优先级任务被低优先级任务长时间阻塞或阻塞一段不确定时间的现象，即所谓的优先级反转（Priority Inversion）
- 示例：假设系统中有3个任务，H、M、L
  - H、M因某种原因被阻塞，系统调度L
  - L执行过程中，H转为就绪，H抢占L
  - H执行过程中，要访问L的共享资源，但是L未执行，不能释放；期间M可能转为就绪
  - 系统调度M
  - 结果：M先执行，H等待M完成后才能执行

# 8.3 Task Mangement

- **Problem——Priority Inversion**



# 8.3 Task Mangement

## 优先级反转解决方法

- 优先级继承（Priority Inheritance）
  - 设S为正占用某项共享资源的进程P，以及所有正在等待占用这个资源的进程的集合
  - 找出这个集合中的优先级最高者，其优先级为P’
  - 把进程P的优先级设置成P’
- 优先级封顶（Priority Ceiling，优先级天花板）
  - 设S为所有可能竞争使用某项共享资源的进程集合。首先为S规定优先级上限P’，使得S中所有进程的优先级都小于P’（注意：P’并不一定是整个系统中的最高优先级）
  - 在创建使用资源的信号量时，将P’作为一个参数
  - 每当有进程通过信号量取得共享资源时，将此进程的优先级暂时提高到P’，一直到释放资源时才恢复其原有的优先级

# 8.3 Task Mangement

## ➢ Solution for Priority Inversion

- Priority Inheritance
  - When a high-priority thread attempts to lock a mutex already locked by a lower-priority thread, the Priority Inheritance Protocol (PIP) temporarily *raises the priority of the low-priority thread to match that of the blocked thread* until the low-priority thread unlocks the mutex.
  - Advantage: It is transparent to the application.
  - Disadvantage: Adds complexity to the kernel.

# 8.3 Task Mangement

## Solution for Priority Inversion

- Priority Ceiling
  - Priority of the low-priority thread is *raised immediately when it locks the mutex* rather than waiting for a subsequent lock attempt by a higher-priority thread.
  - Advantage:Easy to implement.
  - Disadvantage: The priority ceiling value must be *predetermined* for use with the mutex; this value must be the highest among all the threads that attempt to lock the same mutex.

# 8.4 Embedded File System

- 嵌入式文件系统与桌面文件系统有较大区别：嵌入式文件系统要为嵌入式系统的设计目的服务，不同用途的嵌入式操作系统下的文件系统在许多方面各不相同。
- 嵌入式Linux常用文件系统：第二版扩展文件系统（Ext2fs）、JFFS和YAFFS
- 目标
  - 使用简单方便
  - 安全可靠
  - 实时响应
  - 接口标注的开放性和可移植性
  - 可伸缩性和可配置性
  - 开放的体系结构
  - 资源有效性
  - 功能完整性
  - 热插拔
  - 支持多种文件类型

# 8.4 Embedded File System

- **Flash Memory上的两种技术**
  - NAND：串行；顺序读取；适合大容量；通常需MTD
  - NOR：并行；随机读取；适合数据或程序存储；XIP
- **Xsbase开发平台上所使用的闪存**
  - Intel StrataFlashMemory 28F128J3A
- **Ext2fs、JFFS和YAFFS**
  - ext、ext2、xia、vfat、minix、msdos、umsdos、proc、smb、ncp、iso9660、sysv、hpfs、affs、ufs、vfs等

# 8.4 Embedded File System

- **第二版扩展文件系统（Ext2fs）的优点**
  - Ext2fs支持达4 TB的内存（Ext是2G）。
  - Ext2fs文件名称最长可以到1012个字符。
  - 当创建文件系统时，管理员可以选择逻辑块的大小（通常大小可选择1024、2048和4096字节）。
  - Ext2fs实现快速符号链接：不需要为此目的而分配数据块，并且将目标名称直接存储在索引节点表中，这使性能有所提高，特别是在速度上。

# 8.4 Embedded File System

- **JFFS和YAFFS**
  - JFFS文件系统主要针对NOR FLASH设计，是一种基于Flash的日志文件系统。
  - JFFS2的底层驱动主要完成文件系统对Flash芯片的访问控制，如读、写、擦除操作。
  - YAFFS主要针对NAND FLASH设计，和JFFS相比它减少了一些功能。自带NAND芯片驱动，并且为嵌入式系统提供了直接访问文件系统的API。
  - YAFFS2是YAFFS的改进版本。

谢 谢！