无线传感器网络
——WSN操作系统

重庆邮电大学

主要内容

- **WSN对操作系统的需求**
- **操作系统的基本概念**
- **TinyOS操作系统**

WSN操作系统

是否有必要为传感器节点配备一个操作系统？

- 选择**1:**直接在硬件上设计传感网应用程序，完全不使用操作系统。—多用于资源极度受限型的**WSN**
- 选择**2**：直接使用现有的嵌入式操作系统，例如**Vxworks**，**UcOS**、**Linux**等。—多用于资源富裕型的**WSN**
- 选择**3**：开发新的专门为传感网进行优化和定制的**WSN**操作系统。 —代表着未来的发展方向

10年前，很难想象全世界大部分手机都会使用同一个操作系统，但软硬件的不断发展和用户应用的需求，使智能手机操作系统（**Android**或**IOS**）成为主流；传感器节点同样是计算设备，下一个**10**年的发展是否将演进同样的技术路线？**WSN**操作系统的霸主又会是哪家？

WSN操作系统带来的好处

- 操作系统本身带来的好处
  - 开发人员不需要直接面向硬件编程，操作系统提供了统一的接口；
  - 操作系统本身会提供很多基础服务，开发人员无需再单独开发，例如进程管理服务、硬件驱动等；
  - 提供软件的重用性，已有的软件成果可以跟随操作系统，应用在不同的平台上。
- 专门为WSN优化的操作系统带来的好处
  - 高效的使用WSN节点的存储、计算资源；
  - 自带很多WSN协议；
  - 能够对并发程度高、执行过程短的逻辑控制流程提供支持；
  - 模块化程度高

WSN操作系统的设计目标

- 主要设计目标是在有限资源约束下，以模块化、可升级的系统结构实现低能耗、高可靠、实时的操作系统功能，支持密集型的并发操作，并支持节点系统的可重构和自适应能力。
- 传感器网络对操作系统的需求（**P78-79**）

WSN操作系统的设计要素

- 微内核
  - 通用操作系统内核和应用程序相互独立，界限分明
  - **WSN**操作系统通常其核心只提供任务调度、中断管理、时钟管理、任务间通信等功能，文件系统、存储管理都作为可选任务
  - 应用程序的代码嵌入到操作系统中，一起编译成一个镜像文件
- 调度
- 内存管理
- 动态重编程

TinyOS操作系统

- **TinyOS系统介绍**
- **TinyOS**是一个典型的无线传感器网络操作系统。由加利福尼亚大学伯克利分校提出，它提出了很多全新的设计概念，能够很好地满足无线传感器网络操作的要求，是国内外使用最广泛的无线传感器网络操作系统。

TinyOS使用的主要技术

- 组件化编程
- 轻量级线程
- 二级调度
- 主动消息通信机制
- 事件驱动模型

阅读第3.7节内容，深入了解各个机制的具体技术细节。

TinyOS基础

- **TinyOS编程语言**
  - TinyOS使用nesC语言编程，使用nesC编写的应用程序是通过把一个或多个组件连接起来从而组成一个完整的可执行程序。nesC有两种类型的组件:模块(module)和配置(configuration)。
- **TinyOS组件模型（分为三类）**
  - 硬件抽象组件：负责物理硬件映射
  - 合成硬件组件：实现不同数据格式交互
  - 高层软件组件：负责数据处理、路由和传输等。
- **TinyOS不支持动态内存分配**

TinyOS调度

- 采用事件和任务两级调度机制
- 任务（Task）作为轻量级线程，按FIFO方式调度，任务之间不允许抢占，非实时
- 事件分为硬件事件和软件事件。硬件事件是底层发出的中断，软件事件则是通过signal关键字来触发中断。
- 硬件事件中断处理线程可以打断任务和低优先级中断线程，故可对硬件中断进行响应。

TinyOS的模拟器

- TOSSIM是TinyOS的模拟器。它通过把组件替换成模拟的实现来向用户提供模拟服务。
- TOSSIM是一个离散的时间模拟器，当它运行的时候，它从时间队列中依次取出事件(以时间排序)并且执行它们。
- TOSSIM是一个程序库。用户必须写一个程序用来配置和运行模拟。TOSSIM支持两种编程接口；Python和C++。

附：回顾操作系统相关知识

回顾内容：操作系统的四大功能

- 进程管理
- 存储管理
- 文件管理
- 设备管理

其中进程管理是操作系统的核心功能，是操作系统与监护程序区分的标志。

进程的定义

- 把一个程序在一个数据集合上的一次执行成为一个进程。
- 进程与程序的区别
  - 程序是静止的，进程是动态的
  - 进程包括程序和程序处理的对象（数据集）
  - 进程能够得到程序处理的结果
  - 火车的例子：程序是火车，进程是列车。

进程的属性

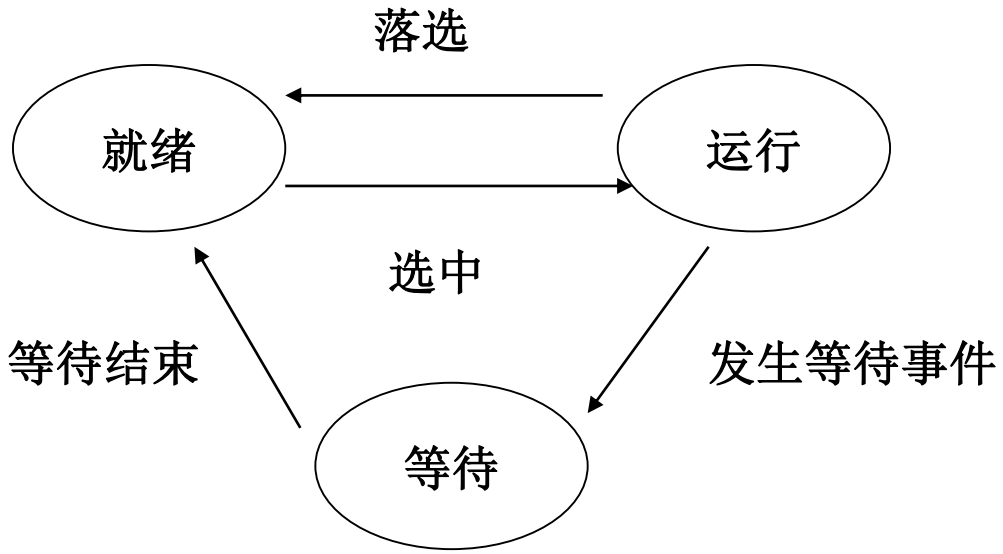
- 进程是动态的
- 多个不同的进程可以包含相同的程序
- 进程可以并发执行
  - 并发的概念：多个进程交替执行，一个进程的工作没有完成之前，另一个进程可以开始工作。这些同时执行的进程是轮流占用处理器的，把它们称为是并发执行的
  - 宏观上是同时执行的，微观上任一时刻单处理器只能运行一个进程
  - 并发可以提高系统的吞吐量

进程的状态

- 进程具有三种状态，一个进程在任何时刻总是处于其中的一种状态。
- 等待态：等待某个事件的完成
- 就绪态：等待系统分配处理器以便运行
- 运行态：占用处理器正在运行

进程状态的转换

- 三态转换图



进程调度

- 利用进程调度算法来解决多个进程竞争处理器使用权的问题。
- 按照某种算法从同时就绪的进程队列中选择一个进程，让它占用处理器来运行。

进程调度的基本算法

- 先来先服务调度算法
- 优先级调度算法
  - 非抢占式(不可剥夺）
  - 抢占式（可剥夺）
  - 优先级的确定方式
- 时间片轮转调度算法
  - 时间片大小的确定
  - 不同的进程可使用不同长度的时间片

分时与实时

- 通用操作系统一般采用以时间片为基础的综合调度算法，Windows和Unix、Linux都属于分时操作系统
- 嵌入式操作系统一般采用基于优先级的调度算法，具有强实时性。
- 两者设计的不同是由于应用领域的不同所导致的。

线程的概念

- 线程是进程中可独立执行的子任务。
- 一个进程中可以有一个或多个线程，每个线程都有一个唯一的标识符
- 举例：同一个网络程序进程，可以有收、发、数据处理三个线程。

线程与进程的联系

- 线程与进程有许多相似之处，往往把线程成为轻量级进程（lightweight process）
- 进程与线程的根本区别在于：
  - 每个进程都有自己的主存空间，而同一进程的各个线程是共享该进行的主存空间的，进程中的所有线程对进程的整个主存空间都有存取权限
  - 进程是资源分配单位，线程是调度和执行单位

任务，进程与线程

- 三个概念在嵌入式开发过程中频繁出现，易令人混淆
- 任务 Task，进程 Process，线程：Thread
- 任务一般用于嵌入式操作系统中。进程和线程多用于通用操作系统，但也广泛用于嵌入式操作系统
- 任务的概念一般相当于线程，但在某些场合，进程也被称为任务。例如，Intel的技术资料中就把Unix/Linux下的进程称为Task
- 例如，嵌入式操作系统μC/OS-II中对任务的定义：一个任务，是一个简单的程序，该程序可以认为CPU完全只属该程序自己。每个任务都是整个应用的某一部分，每个任务被赋予一定的优先级，有它自己的一套CPU寄存器和自己的栈空间。典型地、每个任务都是一个无限的循环。

Thank you !