μC/OS-II

- 透彻的源码分析
- 详尽的原理介绍
- 经典的移植案例
- 简明的协议栈设计

标准教程

杨宗德 张兵 编著

μC/OS-II

人民邮电出版社
POSTS & TELECOM PRESS

μC/OS-II

标准教程

杨宗德 张兵 编著

μC/OS-II

人民邮电出版社
北京

图书在版编目（CIP）数据

μC/OS-Ⅱ标准教程 / 杨宗德，张兵编著. —北京：人民邮电出版社，2009.5
ISBN 978-7-115-20442-4

Ⅰ. μ… Ⅱ. ①杨…②张… Ⅲ. 实时操作系统－教材
Ⅳ. TP316.2

中国版本图书馆CIP数据核字（2009）第025927号

内 容 提 要

本书主要介绍当前最新版本的μC/OS-II（2.80版本）实时操作系统，包括内核分析及其在ARM 9内核处理器（S3C2410）上的移植方法。内核方面主要包括μC/OS-II操作任务管理、任务级任务调度和中断级任务调度、系统启动与初始化、时钟任务管理、任务间通信机制、任务间单事件和多事件同步机制、内存管理方式。本书还通过具体实例介绍μC/OS-II系统在以ARM 9为内核的嵌入式处理器S3C2410上的移植方法，最后对轻量级TCP/IP协议栈——μC/TCP-IP协议栈进行了概要介绍。

本书可以作为高等院校学习嵌入式操作系统原理的专业教材，也适合有意从事嵌入式系统开发的工程技术人员阅读。本书假定读者有较好的C语言基础和数据结构基础知识，如果读者对ARM处理器有一定的了解，将更容易掌握本书内容。

μC/OS-Ⅱ标准教程

◆ 编　　著　杨宗德　张　兵
责任编辑　刘　浩

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：787×1092　1/16
印张：17.5
字数：416千字　　2009年5月第1版
印数：1－4 000册　　2009年5月北京第1次印刷

ISBN 978-7-115-20442-4/TP

定价：39.00元

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

随着信息技术的发展，嵌入式技术也已经广泛运用到日常生活中的方方面面，嵌入式系统在消费、汽车电子、微控制、无线通信、数码产品、网络设备、安全系统等领域的应用方兴未艾。从广义概念来说，除了通用的计算机系统应用外，其他所有的智能电子设备都属于嵌入式系统。从狭义概念来讲，嵌入式系统主要有以下两个重要特征：

- 在硬件上，嵌入式系统至少拥有一个高性能处理器作为硬件平台（目前以32位处理器为主流），如ARM、MIPS系列处理器；
- 在软件上，嵌入式系统以一个多任务操作系统为软件开发平台，如Linux、Windows CE、Symbian、μC/OS-II、VxWorks等。

随着越来越多的公司、研究单位、大专院校以及个人开始进行嵌入式系统的研究，嵌入式系统开发与设计将是未来相当长一段时间内电子领域研究的热点。而对于即将从事嵌入式的工程师来说，在嵌入式专业知识体系上，应该至少掌握以下知识点。

- **一种架构的嵌入式处理器**。掌握一种当前流行的32位处理器，如ARM、MIPS系列，对该类型的处理器结构、汇编语言、硬件管理方式及外部端口和协议有较清楚的认识。
- **一种嵌入式实时操作系统**。熟悉一种较简单、开源的嵌入式操作系统的内核，如μC/OS-II、Linux操作系统，熟悉该操作系统的任务管理方式、任务调度方式及资源管理方式。这也是本书主要介绍的内容。
- **一套嵌入式软硬件开发工具**。根据处理器的不同而选择不同的底层程序开发工具（每种类型的处理器有不同的底层程序开发工具，如ARM处理器流行的开发环境为ADS），根据选用的嵌入式操作系统选择应用平台对应的开发工具（如Linux系统平台的开发软件包为GCC套件）。
- **一类嵌入式开发语言**。目前在嵌入式设备上的底层程序（驱动、操作系统、网络）绝大多数都是采用C语言编写的，在上层开发中会采用C++等高级语言，因此建议采用C系列语言为学习对象。

本书主要内容

本书主要介绍μC/OS-II（2.80版本）实时操作系统内核原理及其在ARM 9内核处理器（S3C2410）上的移植方法，包括任务及任务调度、系统启动与初始化、任务间通信与同步、内存管理、操作系统移植和μC/TCP-IP 协议栈的应用，并以具体实例演示各知识点的应用。

第1章介绍实时操作系统的基本概念，主要涉及实时操作系统的设计原则、实时操作系统的分类以及本书的研究及学习办法。

第 2 章介绍μC/OS-II 操作系统事务管理基本单元——任务管理，主要涉及任务管理的基本方式，任务的基本属性及任务管理操作。

第 3 章介绍μC/OS-II 操作系统任务的调度策略，主要涉及任务级任务调度和中断级任务调度的方法。

第 4 章介绍μC/OS-II 操作系统启动与时钟任务管理，即μC/OS-II 在启动前的准备工作，同时，就系统运行的第 1 个任务——时钟任务进行介绍。

第 5 章介绍μC/OS-II 操作系统任务间通信机制：消息队列及消息邮箱，这两种机制主要实现任务间数据的传递。消息队列可以同时传送多个消息，而消息邮箱每次只能传递一个消息。

第 6 章介绍μC/OS-II 操作系统任务间的单事件同步机制：信号量与互斥锁。信号量是用来标识可用资源的个数，而互斥锁则是对资源进行互斥访问的机制。

第 7 章介绍μC/OS-II 操作系统多事件同步机制：事件组标志。事件组标志用于管理任务间多个事件的同步。

第 8 章介绍μC/OS-II 操作系统内存分区管理，即管理任务在执行过程中申请的堆空间。

第 9 章介绍μC/OS-II 操作系统在 ARM9 内核处理器（S3C2410）上的移植过程，主要涉及到存储空间规划、中断处理、任务调度部分源代码修改和对标准 C 函数库的移植。

第 10 章概要介绍了轻量级 TCP/IP 协议栈——μC/TCP-IP 协议栈的设计原理，为读者学习网络编程打下基础。

本书附录就学习本书代码所使用的编译工具 Visual C++ 6.0 集成开发环境和 ADS 集成开发环境、所使用的源代码阅读工具 Source Insight 进行了简要说明。

本书读者对象

本书可以作为高等院校学习嵌入式操作系统原理的专业教材，也适合有意从事嵌入式系统开发的工程技术人员阅读。本书假定读者有较好的 C 语言基础和数据结构基础知识，如果读者对 ARM 处理器有一定的了解，将更容易掌握本书内容。

本书编写工作

本书由杨宗德、张兵主编。同时特别感谢何伟、刘兆宏、季建华、刘福刚、赵文革等老师，他们对本书提出了大量宝贵指导意见，另外特别感谢石昀、朱元斌、钱文杰、陈功杰、汪洪、刘超、钟晓媛、刘梨平、石霞等同学试读了本书初稿，为本书相关内容提出了宝贵意见。

由于时间关系，本书难免有疏忽和不足之处，甚至有个别错误之处，恳请相关专家批评赐教（电子函件 book_better@sina.com）。

编　者

2009.3

目　录

第 1 章　μC/OS-II 与嵌入式实时操作系统 ……1

1.1　实时操作系统概述 ……2
1.1.1　嵌入式系统软件结构 ……2
1.1.2　实时操作系统内核概述 ……3
1.1.3　常见实时操作系统简介 ……4
1.2　μC/OS-II 内核源代码文档结构 ……6
1.2.1　构建μC/OS-II 模拟编程环境 ……6
1.2.2　测试程序源代码说明 ……8
1.3　μC/OS-II 基本概念 ……9
1.3.1　嵌入式应用程序开发模式 ……9
1.3.2　可重入函数与不可重入函数 ……10
1.3.3　μC/OS-II 临界状态管理 ……11
1.4　小结 ……12
1.5　习题 ……13

第 2 章　μC/OS-II 任务管理 ……15

2.1　案例引入：基于μC/OS-II 的多任务管理 ……16
2.1.1　μC/OS-II 多任务示例运行结果 ……16
2.1.2　μC/OS-II 多任务代码分析 ……16
2.2　μC/OS-II 任务基本属性 ……17
2.2.1　C 语言可执行代码结构 ……17
2.2.2　μC/OS-II 任务结构 ……18
2.2.3　μC/OS-II 任务栈 ……19
2.2.4　μC/OS-II 任务控制块 ……21
2.2.5　μC/OS-II 任务优先级 ……26
2.2.6　μC/OS-II 任务状态 ……26
2.2.7　系统任务 ……28
2.3　μC/OS-II 任务管理函数源码分析 ……32
2.3.1　创建任务 ……32
2.3.2　初始化任务栈 ……35
2.3.3　初始化任务控制块 ……37
2.3.4　扩展创建任务 ……40
2.3.5　删除任务 ……42
2.3.6　请求删除任务 ……45
2.3.7　挂起任务 ……47
2.3.8　恢复任务 ……49
2.3.9　设置任务名称 ……50
2.3.10　获取任务名称 ……52
2.3.11　读取任务 TCB 信息 ……53
2.4　应用实例：多任务管理应用分析 ……55
2.4.1　基本功能 ……55
2.4.2　程序实现及源码分析 ……55
2.5　小结 ……58
2.6　习题 ……58

第 3 章　μC/OS-II 任务调度与系统初始化 ……59

3.1　μC/OS-II 任务级任务调度机制 ……60
3.1.1　μC/OS-II 调度算法 ……60
3.1.2　μC/OS-II 任务就绪表 ……60
3.1.3　获取最高优先级就绪任务 ……62
3.2　μC/OS-II 任务级任务调度 ……63

3.2.1 任务级任务调度算法分析……63
3.2.2 任务级任务切换 OS_TASK_SW()……64
3.2.3 调度器上锁与解锁……66
3.2.4 修改任务优先级……67
3.3 μC/OS-II 中断级任务调度……69
3.3.1 μC/OS-II 中断管理……69
3.3.2 IRQ 中断处理过程及中断级调度……70
3.3.3 OSIntEnter()进入中断管理函数……72
3.3.4 OSIntExit()退出中断管理函数……72
3.3.5 中断级任务调度切换函数 OSIntCtxSw()……73
3.3.6 FIQ 中断处理过程及中断级调度……74
3.4 任务级任务调度实例……76
3.4.1 程序功能……76
3.4.2 程序实现及源码分析……76
3.5 小结……78
3.6 习题……79

第 4 章 μC/OS-II 系统启动与时钟任务管理……81

4.1 μC/OS-II 系统启动过程分析……82
4.1.1 μC/OS-II 应用程序开发模式……82
4.1.2 OSInit()函数初始化分析……82
4.1.3 OSStart()函数启动系统分析……87
4.1.4 运行最高优先级任务……87
4.2 时钟任务与时钟管理……88
4.2.1 创建系统时钟任务……88
4.2.2 时钟中断服务程序 OSTimeTick()……89
4.2.3 系统时间管理函数源码分析……91
4.3 小结……95
4.4 习题……95

第 5 章 μC/OS-II 任务间通信机制……97

5.1 μC/OS-II 事件管理机制……98
5.1.1 事件控制块……98
5.1.2 事件控制块管理……99
5.2 单一消息传递事件：消息邮箱……101
5.2.1 消息邮箱基本原理……101
5.2.2 创建消息邮箱……101
5.2.3 阻塞式读取消息……103
5.2.4 非阻塞式读取消息……106
5.2.5 发送消息到消息邮箱……107
5.2.6 按指定方式发送数据到消息邮箱……110
5.2.7 删除消息邮箱……111
5.2.8 获取消息邮箱基本信息……114
5.2.9 消息邮箱应用实例……115
5.3 多消息传递事件：消息队列……117
5.3.1 消息队列基本原理……117
5.3.2 创建消息队列……119
5.3.3 发送消息到队列尾……121
5.3.4 发送消息到队首……122
5.3.5 按指定方式发送消息……124
5.3.6 阻塞式读取消息……125
5.3.7 非阻塞式读取消息……128
5.3.8 删除消息队列……129
5.3.9 获取消息队列信息……132
5.3.10 清理消息队列空间……133
5.3.11 消息队列应用实例……134
5.4 小结……137
5.5 习题……138

第 6 章 μC/OS-II 任务间单事件同步机制……139

6.1 任务同步机制：信号量……140

6.1.1 信号量基本原理 ……140
6.1.2 创建信号量 ……140
6.1.3 阻塞式获取信号量 ……141
6.1.4 非阻塞式获取信号量 ……143
6.1.5 释放信号量 ……144
6.1.6 删除信号量 ……145
6.1.7 查询信号量信息 ……147
6.1.8 设置信号量的值 ……148
6.1.9 信号量实现生产—消费问题应用实例 ……149
6.2 互斥事件管理机制：互斥锁 ……153
6.2.1 互斥锁与优先级反转 ……153
6.2.2 创建互斥锁 ……156
6.2.3 阻塞式获取互斥锁 ……158
6.2.4 非阻塞式获取互斥锁 ……161
6.2.5 释放互斥锁 ……162
6.2.6 删除互斥锁 ……164
6.2.7 获取互斥锁基本信息 ……167
6.3 小结 ……168
6.4 习题 ……169

第 7 章 μC/OS-II 多事件同步机制 ……171

7.1 事件组标志同步机制基本原理 ……172
7.1.1 事件组标志基本原理 ……172
7.1.2 初始化事件标志组控制块 ……174
7.2 事件组标志基本操作 ……175
7.2.1 创建事件组标志 ……175
7.2.2 阻塞式等待事件组标志 ……176
7.2.3 非阻塞式等待事件组标志 ……182
7.2.4 修改事件组标志状态 ……184
7.2.5 删除事件标志组 ……187
7.2.6 获取任务就绪标志 ……190
7.2.7 查询事件组标志信息 ……190
7.2.8 获取事件组标志名称 ……191
7.2.9 设置事件组标志名称 …… 192
7.3 使用事件组标志实现读写锁功能案例 …… 193
7.3.1 案例功能及原理说明 …… 193
7.3.2 程序源代码分析 …… 195
7.4 小结 …… 197
7.5 习题 …… 197

第 8 章 μC/OS-II 内存分区管理 …… 199

8.1 内存分区管理基本原理 …… 200
8.1.1 μC/OS-II 内存分区管理机制 …… 200
8.1.2 内存分区控制块数据空间初始化 …… 201
8.2 μC/OS-II 内存分区管理操作 … 202
8.2.1 创建内存分区 …… 202
8.2.2 申请一个内存分区块 …… 204
8.2.3 释放内存分区块 …… 205
8.2.4 查询内存分区基本信息 …… 206
8.2.5 读取/设置内存分区名称 …… 207
8.3 μC/OS-II 内存管理应用实例 … 210
8.3.1 应用程序基本功能 …… 210
8.3.2 应用程序源代码分析 ……211
8.4 小结 …… 212
8.5 习题 …… 212

第 9 章 μC/OS-II 在 S3C2410 处理器上的移植案例分析 …… 213

9.1 ARM 可执行文件结构分析 …… 214
9.1.1 可执行文件结构分析 …… 214
9.1.2 ADS 下可执行文件的编译连接过程分析 …… 215
9.1.3 移植第一步：设置代码存储加载位置 …… 217
9.2 移植第二步：编写系统启动代码 …… 218

9.2.1 开始执行，禁止中断 ····219
9.2.2 初始化栈空间 ····219
9.2.3 复制异常代码与异常向量地址 ····220
9.3 移植第三步：中断处理与时钟中断任务 ····221
9.3.1 S3C2410 中断向量 ····221
9.3.2 S3C2410 中断处理硬件结构 ····223
9.3.3 移植时对中断的处理 ····225
9.3.4 示例：μC/OS-II 系统时钟中断任务管理 ····228
9.4 移植第四步：修改与调度相关的汇编代码 ····229
9.4.1 临界状态问题 ····229
9.4.2 数据类型问题 ····230
9.4.3 任务调度问题与钩子函数问题 ····231
9.5 移植后续工作：添加硬件驱动程序 ····232
9.5.1 重新写 C 函数以实现信息从串口输出 ····232
9.5.2 实现串口驱动程序 ····234
9.5.3 重写堆栈空间初始化函数 ····237
9.6 小结 ····238
9.7 习题 ····238

第 10 章 μC/TCP-IP 协议栈设计分析 ····239

10.1 TCP/IP 协议栈概述 ····240
10.1.1 标准 TCP/IP 协议栈模型 ····240
10.1.2 BSD 面向连接的 TCP 通信编程过程 ····241
10.1.3 BSD 面向无连接的 UDP 通信实现 ····242
10.1.4 轻量级 TCP/IP 协议栈和标准 TCP/IP 协议栈比较 ····242
10.1.5 μC/TCP-IP 协议栈介绍及特点 ····243
10.2 μC/TCP-IP 协议栈设计 ····243
10.2.1 μC/TCP-IP 协议栈基本框架 ····243
10.2.2 进程基本形式 ····245
10.2.3 接收数据过程 ····245
10.2.4 发送数据过程 ····246
10.3 μC/TCP-IP 协议栈实现 ····247
10.3.1 IP 协议栈设计及实现 ····247
10.3.2 TCP 协议设计及实现 ····249
10.3.3 UDP 协议设计及实现 ····252
10.3.4 ICMP 协议设计及实现 ····253
10.3.5 ARP 设计及实现 ····255
10.3.6 缓冲区设计及管理 ····257
10.4 小结 ····260
10.5 习题 ····260

附录 编译程序工具简介 ····261

A.1 Visual C++ 6.0 集成开发环境 ····261
A.2 ADS 集成开发环境 ····264
A.3 Source Insight 源代码查看工具 ····269

μC/OS-II

第1章 μC/OS-II与嵌入式实时操作系统

随着信息技术的发展，嵌入式产品已经被广泛运用到日常生活中的各个方面。关于“嵌入式系统”的定义有很多种说法，目前最通用的定义为：“以应用为中心，以计算机技术为基础，软件硬件可裁减，功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。”作者曾经给予了一个更为广泛的概念：“除通用计算机系统外，一切智能的电子设备都属于嵌入式设备。”

在此，读者并没有必要一直追究嵌入式系统的严格定义，因为这没有太多意义。不过，嵌入式工程师们更强调系统的概念，他们通常认为狭义的嵌入式系统需要涵盖软件和硬件两个方面，与以前简单的单片机系统有着本质的区别。

- 在硬件上，嵌入式系统至少拥有一个高性能处理器（目前以32位处理器为主流）作为硬件平台，如ARM、MIPS等处理器。
- 在软件上，嵌入式系统拥有一个实时操作系统作为软件系统平台，如Linux、WinCE、Symbian、μC/OS-II、VxWork等。

应用于嵌入式设备的实时操作系统被称为嵌入式实时操作系统，即RTOS（实时操作系统），其最显著的特点是实时性。因为常见的嵌入式系统都是实时系统，因此有时候将实时系统和嵌入式操作系统混合使用。

本章第1节主要介绍嵌入式实时操作系统的基本概念、内核类型及设计原则，并简要介绍常见的嵌入式操作系统。

本章第2节主要介绍μC/OS-II操作系统源代码基本文档结构、本书研究方法及应用示例源代码结构。

本章第3节主要介绍为便于后续章节学习所需要的基本概念，包括软件开发模式、可重入函数及临界状态等。

1.1 实时操作系统概述

1.1.1 嵌入式系统软件结构

图 1-1 所示为嵌入式软件系统基本模型。在一个嵌入式系统中，软件是具体功能的逻辑实现。根据应用需要，一个嵌入式系统有可能包含板级支持包、实时操作系统、文件系统、图形用户界面、系统管理接口和应用程序等部分。当然，并不是所有嵌入式设备的软件系统都完全遵循这一模型。在某些具体应用中，有可能只需要其中的几个组件，也有可能将其中的几个组件组合在一起，或者增加几个部分。然而，对于大多数嵌入式设备来说，采用这种层次结构来开发整个系统的软件具有很强的可操作性和可维护性。

<table>
<tr><td colspan="4">应用程序层（Application）</td></tr>
<tr><td colspan="2">文件系统（FS）</td><td>图形界面（GUI）</td><td>系统管理接口</td></tr>
<tr><td colspan="4">实时操作系统（RTOS）</td></tr>
<tr><td colspan="3">板级支持包（BSP）</td><td>固件</td></tr>
<tr><td colspan="4">硬件层</td></tr>
</table>

图 1-1　嵌入式软件系统基本模型

硬件层是整个系统的硬件组成部分，其典型特点是至少拥有一个 32 位微处理器。

固件是指一些处理器中包含的一段微代码，如一个串口 RS232 驱动。当然，并不是所有的处理器都包含固件，其在很大程度上类似于通用计算机中的 BIOS。

板级支持包（Board Support Packet），在很多地方又被称为 BootLoader。使用过 Linux 操作系统的读者应用知道 GRUB（启动 Linux 操作系统的引导程序）。GRUB 是一个应用在通用计算机上的 BootLoader 程序，只是嵌入式设备中所采用的 BootLoader 不同而已。BootLoader 主要完成的功能是引导操作系统。

实时操作系统是整个嵌入式软件系统的核心。实时操作系统与通用操作系统最显著的区别在于对中断响应、可抢占性和裁减性等方面。目前，嵌入式实时操作系统主要有 Linux、WinCE、μC/OS、Symbian、VxWorks 等。实时操作系统的内核一般都很小，例如一个被裁减了的 Linux 操作系统的内核仅 500KB 左右。

文件系统是存储文件的基础，常见的文件系统有 FAT32、EXT3 等。在嵌入式设备中，需要使用专门的文件系统，这是因为嵌入式设备对可靠性的要求很高。

如果系统需要开发图形用户界面，则需要图形用户界面接口。目前，不同的操作系统有不同的图形界面接口，如嵌入式 Linux 下的图形界面主要有 MiniGUI 和 QT。

应用软件层，即用户设计的、针对特定应用的应用软件。在开发应用软件时，可以用到底层提供的大量函数，包括操作系统提供的大量系统调用。

是否拥有实时操作系统是衡量一个嵌入式系统最为重要的参数之一，然而，嵌入式处理器架构与传统的 X86 计算机架构有很大的差异，也就是说，原有的、应用于 X86 平台的操作系统不能被直接运用到嵌入式设备中。

因此，为了使原有的操作系统能够在采用新的处理器架构的嵌入式设备中得以运用，则需要对操作系统内核进行修改。这一过程也就是移植，这是当前嵌入式软件开发中最为重要的内容之一，也是难度最大的一部分，同时也是本书介绍的重点。

1.1.2 实时操作系统内核概述

作为一种嵌入式操作系统，嵌入式实时操作系统具有嵌入式软件共有的可裁减、低资源占用、低功耗等特点；而作为一种实时操作系统，它与通用操作系统（如 Windows、UNIX、通用 Linux 等）相比有很大的差别。

实时操作系统所遵循的设计原则是采用各种算法和策略，始终保证系统行为的可预测性，即在任何情况下，在系统运行的任何时刻，操作系统的资源配置策略都能为争夺资源（包括 CPU、内存、网络带宽等）的多个实时任务合理地分配资源，使每个实时任务的实时性要求都能得到满足。

因此，实时操作系统主要关注的不是系统的平均表现，而是要求在最坏情况下也能满足每个任务的实时性要求。也就是说，实时操作系统更关注个体表现，特别是个体在最坏情况下的表现，即保证在任何情况下都能够满足任何一个任务的实时性要求。

根据内核是否可以被剥夺（抢占），实时操作系统内核亦可分为不可剥夺型内核和可剥夺型内核，两者的区别在于任务是否在任何时刻都可以被抢占。

1．不可剥夺型内核

图 1-2 所示为不可剥夺型内核中断响应示意图，其基本过程如下。

（1）就绪任务 A 拥有 CPU 的控制权，开始执行。

（2）某外部事件引起中断，CPU 暂停 A 的执行，假定中断响应时使更高优先级的任务 B 就绪。

（3）中断返回，继续执行任务 A，即不抢占 A 的执行权利。

（4）任务 A 执行完毕，系统执行新的调度。如果 B 此时优先级最高，方执行任务 B。

2．可剥夺型内核

图 1-3 所示为可剥夺型内核中断响应示意图，其基本过程如下。

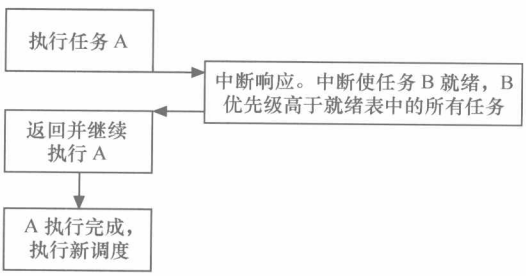

图 1-2　不可剥夺型内核中断响应示意

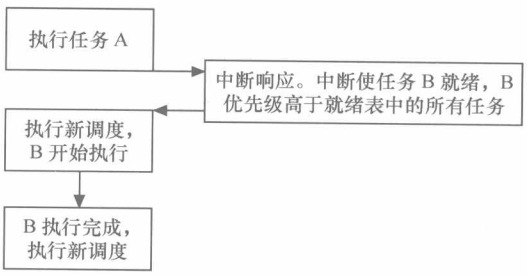

图 1-3　可剥夺型内核中断响应示意

（1）就绪任务 A 拥有 CPU 的控制权，开始执行。

（2）某外部事件引起中断，CPU 暂停 A 的执行，假定中断响应时使更高优先级的任务 B 就绪。

（3）中断返回，因 B 优先级高于 A，故中断任务 A 的执行，并执行新的调度，使任务 B 拥有 CPU 的控制权。

（4）任务 B 开始执行。如果没有中断使更高优先级的任务就绪，则 B 得以执行完毕；否则将中断 B，响应更高优先级的任务。

（5）任务 B 执行完毕，如果没有比 A 优先级更高的任务就绪，则返回执行 A。

由此可知，两者的主要区别在于中断返回时是否执行新的调度：可剥夺型内核在中断返回时执行新的调度，而不可剥夺型内核则是在任务执行完成后执行新的调度。

1.1.3 常见实时操作系统简介

1. VxWorks 操作系统

VxWorks 操作系统是美国 WindRiver 公司于 1983 年设计开发的一种嵌入式实时操作系统，具有良好的持续发展能力。高性能的内核及友好的用户开发环境，使其在嵌入式实时操作系统领域牢牢占据着一席之地。其主要组成部分有微内核、I/O 子系统、文件系统、网络系统等。VxWorks 嵌入式操作系统具有以下显著特点。

- 销售额最大的实时操作系统，价格昂贵。
- 通常只提供二进制码内核。
- 支持多种 CPU。
- 完整的开发工具和测试工具。
- 完备的设备驱动程序和应用模块。
- 技术支持需付费。
- 支持 POSIX 标准。
- 需要中等硬件系统资源，性能优越，功能齐全。
- 可靠性、实时性和可裁减性标准。
- 支持多种处理器，如 X86、i960、Sun Sparc、Motorola MC68xxx、MIPS、POWER PC 等。
- 大多数的 VxWorks API 是专有的，例如火星机器人 API。

2. WinCE 操作系统

Windows CE（简称 WinCE）是世界上最大的操作系统提供商——微软设计的、应用于嵌入式设备上的专用操作系统。在 Windows 操作系统的影响下，Windows CE 已经在嵌入式产品中占有了相当广阔的市场。目前，Windows CE 在娱乐设备、数字机顶盒、数字医疗终端设备等领域得到了广泛的应用。Windows CE 5.0 是一种针对小容量、移动式、智能化、32 位、连接设备的模块化实时嵌入式操作系统，主要有以下特点。

（1）开发环境全面。其提供的 Windows Builder 为开发人员提供了迅速建立基于 Windows CE 的嵌入式系统所需要的各种工具，Platform Builder 的集成开发环境（IDE）允许用户设计、建立、测试、调试所需要的 WinCE 操作系统。

（2）模块化。Windows CE 使嵌入开发人员能够对设备进行定制，从而加快了系统开发过程，提高了开发速度。

（3）Windows CE 属于软实时操作系统，支持嵌套中断及 256 个优先级。

（4）强大的多媒体处理能力。Windows CE 提供 DirectDraw、DirectSound 及 DirectShow 3 个 API 函数用于开发多媒体业务。

（5）强大的网络和通信能力。Windows CE 提供强大的通信和网络互连能力，因此可以将嵌入式设备与 Windows 设备、网络设备及其他设备进行互连。

3. Symbian 操作系统

Symbian 操作系统是由 Symbian 公司研发的、专用于智能手机的嵌入式操作系统。

Symbian 公司是一家由摩托罗拉、西门子、诺基亚等几家大型移动通讯设备制造商共同出资组建的合资公司，专门研发手机操作系统（其名亦为 Symbian）。Symbian 操作系统在智能移动终端上拥有强大的应用程序及通信处理能力，这归功于它有一个非常健全的核心：强大的对象导向系统、企业用标准通信传输协议及完美的 SUN Java 语言支持。

Symbian 操作系统提供了灵活的用户界面（UI）框架，不但使开发者得以快速掌握必要的技术，同时还使手机制造商能够推出不同界面的产品。Symbian 操作系统手机可以采用多种用户界面形式主要分为以下 3 大类：Symbian s60、Symbian UIQ 和 Symbian s80。

4．RT-Linux 操作系统

Linux 由 UNIX 操作系统发展而来，其内核由网络上组织松散的黑客队伍一起从零开始编写而成。Linux 遵循 GNU GPL（通用公共许可证）协议，由于不排斥开发商对自由软件进行进一步开发，也不排斥在 Linux 上开发商业软件，故而出现了很多 Linux 发行版，如 Slackware、Redhat、TurboLinux、OpenLinux 等十多种，而且数量还在增加。还有一些公司在 Linux 上开发商业软件或把其他 UNIX 平台的软件移植到 Linux 上来，如今很多 IT 界的大腕，如 IBM、Intel、Oracle、Infomix、Sysbase、Netscape、Novell 等公司都宣布支持 Linux。开发商的加盟弥补了纯自由软件的不足，从而使得 Linux 得以迅速普及。

Linux 是开放源码的，不存在黑箱技术，其内核小、功能强大、运行稳定、效率高，易于定制剪裁，且在价格上极具竞争力。Linux 不仅支持 X86 CPU，还可以支持其他数十种 CPU 芯片。Linux 具有以下特点。

（1）内核精简，性能高，运行稳定。

- 良好的多任务支持。
- 适用于不同的 CPU 体系架构，如 X86、ARM、MIPS、ALPHA、SPARC 等。
- 可伸缩的结构使 Linux 适合于各种简单或复杂的嵌入式应用。

（2）外设接口统一。

- 以设备驱动程序的方式为应用提供统一的外设接口。
- 开放源码，软件资源丰富。广泛的软件开发者的支持，价格低廉，结构灵活，适用面广。
- 完整的技术文档，便于用户的二次开发。

（3）可以定制网络支持，具备完整的 TCP/IP 栈，同时支持大量其他的网络协议。

（4）可定制的文件管理系统，包括 NFS、EXT2、FAT16/32 等。

Linux 是一款目前最为流行的、开放源代码的操作系统，从 1991 年问世至今，无论是在 PC 平台上，还是在嵌入式应用上都大放光芒，逐渐形成了与其他商业嵌入式操作系统抗衡的局面。目前正在开发的嵌入式项目中有 70%选择 Linux 作为嵌入式操作系统。

5．μC/OS 操作系统

μC/OS 系统由美国人 Jean Labrosse 于 1992 年完成，1998 年发展到 μC/OS-II，目前的版本为 μC/OS -II V2.83。2000 年，μC/OS 得到美国航空管理局（FAA）的认证，可以用于飞行器中，其开发网站为 www. μC/OS -II.com（www.micrium.com）。作为一款典型的嵌入式操作系统，μC/OS-II 系统的应用面非常广泛，如照相机、医疗器械、音响设备、发动机控制、高速公路电话系统、自动提款机等。

μC/OS-II 读做“micro COS 2”，意为“微控制器操作系统版本 2”。μC/OS 不但提供了一个完整的嵌入式实时内核的源代码，而且对这些代码的细节进行了详尽的解释。它不仅告诉读者该实时内核是怎么写的，还解释了为什么要这样写。而商业应用中的实时操作系统软件不但价格昂贵，而且其中很多都是所谓的“黑盒子”，即不提供源代码。

μC/OS 源代码的绝大部分是用 C 语言编写的，经过简单的编译，读者就能在 PC 机上运行。由于用汇编语言编写的部分只有 200 行左右，因此该实时内核可以被方便地移植到几乎所有的嵌入式应用类 CPU 上。

到目前为止，μC/OS 已被移植到许多嵌入式微处理器上，如 Analog 设备公司的 AD21xx 系列，ARM 公司的 ARM 6、ARM7 系列，日立公司的 64180、H8/3xx、SH 系列，Intel 公司的 80x86（Real andPM）、Pentium、Pentium II、8051、8052、MCS-251、80196、8096 系列，三菱公司的 M16 和 M32 系列，摩托罗拉公司的 PowerPC、68K、CPU32、CPU32+、68H11、68HC16 系列，飞利浦公司的 XA 系列，西门子公司的 80C166 和 TriCore 系列，TI 公司的 TMS320 系列，以及 Zilog 公司的 Z80 和 Z180 系列等。

1.2 μC/OS-II 内核源代码文档结构

本节将详细介绍μC/OS-II 内核源代码文档结构，并为读者介绍如何建立 Microsoft VC++ 6.0 下的μC/OS-II 测试开发环境。这一节是全书的基础，后续章节中使用到的大量代码都可以直接在 Microsoft VC++ 6.0 下编译测试。

需要强调的是，本书中的测试代码是借助 Windows 平台上的一个模拟系统，而不是真正的嵌入式系统，仅仅是为了读者学习研究的需要而设置的。

1.2.1 构建μC/OS-II 模拟编程环境

本小节首先展示μC/OS-II 操作系统应用程序示例。读者可以在随书光盘中的 ch01 目录结构下查找到相应的源代码，其文件夹名为 01_μC/OS -II_Win32_ templet（可以将此源代码文件拷贝到硬盘上进行测试）。此项目系统环境如下。

- 操作系统：Windows 2003（其他操作系统亦可）。
- 开发环境：Microsoft Visual C++ 6.0。
- μC/OS-II 源代码：V2.80 版本。

如果读者的当前系统没有安装 Microsoft Visual C++ 6.0 开发环境，请事先安装此软件，因为本书中的大量源代码程序都是在此软件开发环境中开发的。

1．源代码文件目录结构

图 1-4 所示为此示例程序文档目录结构。

01_μC/OS -II_Win32_ templet 文件夹中包含本示例应用程序源代码，包括 app.c、app_cfg.h、includes.h 和 os_cfg.h。

μC/OS -II 文件夹中包含μC/OS-II V2.80 源代码程序，此文件夹下各文件的源代码是本书研究的重点。

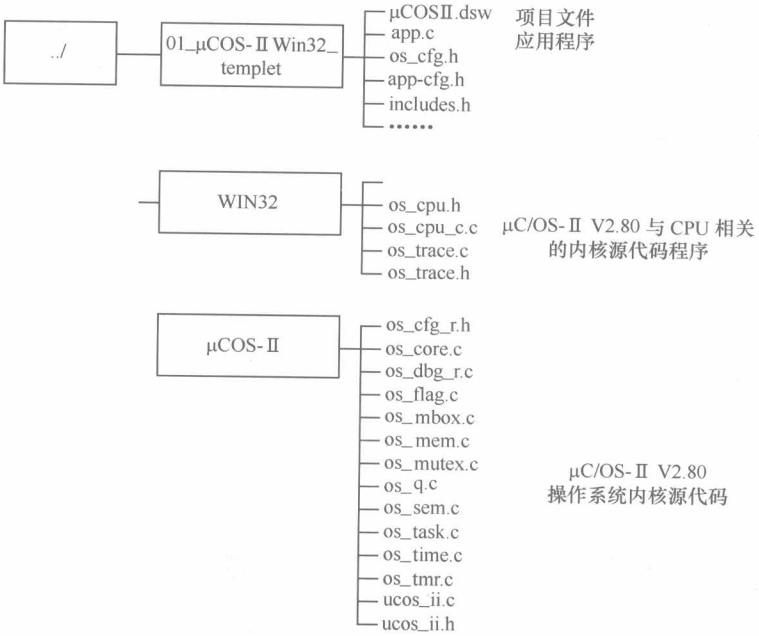

图 1-4　示例程序文档目录结构

WIN32 文件夹中包含与 CPU 硬件相关的源代码程序。由于处理器架构（例如 X86 处理器、ARM 处理器、PowerPC 处理器等）不一样，操作系统与硬件直接相关的源代码也会不一样。本示例程序为了便于初学者学习实时操作系统而选用了 X86 架构源代码程序，本书在后续章节将介绍基于 ARM 处理器的μC/OS-II 操作系统内核源代码，显然其与 CPU 直接相关的内核源代码与此处使用的、基于 X86 的源代码是不一样的。这也正是操作系统移植工作主要研究的内容。

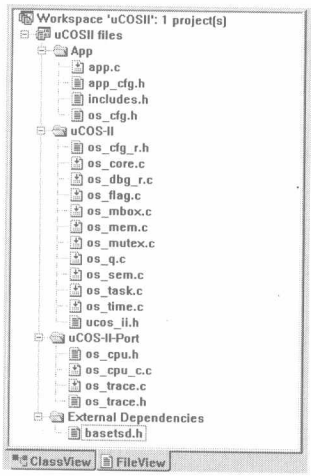

图 1-5　VC++开发环境中的文件列表

2．运行测试结果

因为软件环境配置内容较多，建议第一次使用 Microsoft Visual C++ 6.0 环境开发μC/OS-II 程序的读者，直接使用 Microsoft Visual C++ 6.0 开发环境打开随书光盘中 ch01 文件夹下的 01_μC/OS -II_Win32_ templet \μC/OS II.dsw 项目文件。打开此文件后，Microsoft Visual C++ 6.0 开发环境左侧“FileView”工作面板列出了所有源代码文件夹中的内容，如图 1-5 所示。

如果打开项目文件过程无误，则可在 Microsoft Visual C++ 6.0 环境下单击菜单命令“Build→Rebuild All”重新编译此项目源代码程序。如果所有操作正确，则显示编译正确的提示信息。

```
Deleting intermediate files and output files for project 'μC/OS II - Win32 Debug'.
--------------------Configuration: μC/OS II - Win32 Debug--------------------
Compiling...
app.c
os_core.c
```

```
os_dbg_r.c
os_flag.c
os_mbox.c
os_mem.c
os_mutex.c
os_q.c
os_sem.c
os_task.c
os_time.c
os_cpu_c.c
os_trace.c
Generating Code...
Linking...

μC/OS II.exe - 0 error(s), 0 warning(s)
```

以上信息表示编译成功，并生成了可执行程序μC/OS II.exe。此时在 Microsoft Visual C++ 6.0 环境下单击菜单命令“Build→Execute μC/OS II.exe”将打开图 1-6 所示的、Win32 控制台下的运行结果（按 Ctrl+C 退出）。

```
this is App_one, prio=10,Delay 2 second and start again

this is App_one, prio=10,Delay 2 second and start again
```

图 1-6　运行结果

1.2.2　测试程序源代码说明

本程序仅仅创建一个任务，其功能是每间隔一秒在屏幕上打印一条“this is App_one, prio=10,Delay 2 second and start again”信息。

app.c 文件主要包括两个函数：程序主函数 main()和创建任务函数 App_one()。以下是这两个函数源代码内容的简单说明。

```
#include <includes.h>
#define  TASK_STK_SIZE    128
OS_STK       AppStk_one[TASK_STK_SIZE];

static void App_one(void *p_arg);

void main(int argc, char *argv[])                    //①主函数入口
{
       BSP_IntDisAll();                              //②如果是嵌入式设备，初始化 BSP
       OSInit();                                     //③初始化μC/OS-II 系统，见第 4 章
       OSTaskCreate(App_one,NULL,(OS_STK *)&AppStk_one[TASK_STK_SIZE-1],(INT8U)10);
                                                     //④创建任务，见第 2 章
      OSStart();                                     //⑤启动多任务管理，见第 4 章
}
```

在②行中，此句并没有执行。因为当前运行环境为 X86 处理器上的 Windows 环境，不属于嵌入式设备，因此不需要专门编写系统的初始化代码。如果是嵌入式设备，则此函数调用 BSP 程序初始化嵌入式设备，例如禁止看门狗、禁止所有外部中断功能。③行调用 OSInit()函数初始化系统。在所有μC/OS-II 应用程序中，在创建任何任务或其他内核对象前都需要调用此函数。④行调用 OSTaskCreate()函数创建任务，其具体功能将在 App_one()