第 1 章　概论

20 世纪 60 年代，以晶体管、磁芯存储为基础的计算机系统已开始应用于航空、航天、工业控制等领域，这些系统可看成嵌入式系统的雏形。第一台机载专用数字计算机是奥托内蒂克斯公司为美国海军舰载轰炸机研制的多功能数字分析仪，由几个体积庞大的黑匣子组成，能够进行中央集中处理，开始拥有数据总线雏形。

随后，嵌入式系统处理能力和功能则快速提升。例如，第一款微处理器 Intel 4004 于 1971 年诞生，被广泛应用于计算器和其他小型系统，此时的嵌入式系统已有外存和其他芯片的支持。到 20 世纪 80 年代中期，大多数外部系统芯片集成到一块芯片上作为处理器，称为微控制器（Microcontroller，MCU），也称为单片机，使得嵌入式的应用更为灵活。最早的单片机 Intel 4084 出现在 1976 年。20 世纪 80 年代初，Intel 公司开发出了著名的 8051 单片机，并一直沿用至今。同一时期，Motorola 公司推出了 68HC05，Zilog 公司专门生产 Z80 单片机。这些处理器迅速渗透到家用电器、医疗仪器、仪器仪表、交通运输等领域，带动了嵌入式系统的快速发展。

为了实时处理数字信号，1982 年诞生了首枚数字信号处理芯片（Data Signal Processor，DSP）。现今，已经发展成一类十分重要的多媒体处理芯片。1997 年，来自美国嵌入式系统大会（Embedded System Conference）的报告指出，未来 5 年（从 1997 年算起）仅基于嵌入式计算机系统的全数字电视产品，就将在美国产生一个每年 1500 亿美元的新市场。美国汽车大王福特公司的高级经理也曾宣称，“ 福特出售的‘计算能力’已超过了 IBM ”，由此可以想见嵌入式计算机工业的规模和广度。1998 年在芝加哥举办的嵌入式系统会议上，与会专家一致认为，21 世纪嵌入式系统将无所不在，它将为人类生产带来革命性的发展，实现“ PCs Everywhere ”的生活梦想。

美国嵌入式系统专业杂志 RTC 报道，21 世纪最初的十年中，全球嵌入式系统市场需求量具有比 PC 市场需求量大 10～100 倍的商机。纵观嵌入式技术的发展过程，其出现至今已经有 40 多年的历史，大致经历以下五个阶段。

第一阶段大致在 20 世纪 70 年代之前，可看成嵌入式系统的萌芽阶段，是以单芯片为核心的可编程控制器形式的系统，具有与监测、伺服、指示设备相配合的功能。这类系统大部分应用于一些专业性强的工业控制系统中，一般没有操作系统的支持，通过汇编语言编程对系统进行直接控制。这一阶段系统的主要特点是：系统结构和功能相对单一，处理效率较低，存储容量较小，只有很少的用户接口。由于这种嵌入式系统使用简单、价格低，以前在国内外工业领域应用非常普遍。即使到现在，在简单、低成本的嵌入式应用领域依然大量使用，但已经远不能适应高效的、需要大容量存储的现代工业控制和新兴信息家电等领域的需求。

之后的十多年属于第二阶段，是以嵌入式处理器为基础、以简单操作系统为核心的嵌入式系统。在此阶段，大多数嵌入式系统使用 8 位处理器，不需要嵌入式操作系统支持，其主要特点是：处理器种类繁多，通用性比较弱；系统开销小，效率高；高端应用所需操作系统已达到一定的实时性、兼容性和扩展性；应用软件较专业化，用户界面不够友好。

第三阶段的时段大致是 20 世纪 80 年代末到 90 年代后期，是以嵌入式操作系统为标志的

嵌入式系统，也是嵌入式应用开始普及的阶段。主要特点是：嵌入式操作系统内核小、效率高，具有高度的模块化和扩展性；能运行于各种不同类型的微处理器上，兼容性好；具备文件和目录管理、多任务、网络支持、图形窗口以及用户界面等功能；提供大量的应用程序接口 API 和集成开发环境，简化了应用程序开发；嵌入式应用软件丰富。在此阶段，嵌入式系统的软硬件技术加速发展，应用领域不断扩大。例如，日常生活中使用的手机、数码相机，网络设备中的路由器、交换机等，都是嵌入式系统；一辆豪华汽车中有数十个嵌入式处理器，分别控制发动机、传动装置、安全装置等；一个飞行器上可以有数百个乃至上千个微处理器；一个家庭中也有了几十个嵌入式系统。

第四个阶段从 20 世纪 90 年代末开始，是以网络化和 Internet 为标志的嵌入式系统。随着 Internet 的发展以及 Internet 技术与信息家电、工业控制、航空航天等技术结合日益密切，嵌入式设备与 Internet 的结合将代表嵌入式系统的未来。1998 年 11 月在美国加州圣·何塞举行的嵌入式系统大会上，基于嵌入式实时操作系统的 Embedded Internet 成为一个新的技术热点。

最后一个阶段是 21 世纪初到现在，是以物联网、云计算和智能化为标志的嵌入式系统，也是多核芯片技术、无线技术、互联网发展与信息家电、工业控制、航空航天等技术结合的必然结果。从应用角度而言，移动互联网设备是嵌入式产品的热点，目前，具备网络互联功能的智能终端出货量将达到 4 亿部，比同时期笔记本电脑和台式计算机出货量的总和还多。无处不在的嵌入式系统（智能手机、无线传感器网络、RFID 电子标签等）遍布在人们周围，为人们提供方便快捷的服务。

综上所述，嵌入式系统技术正在日益完善，高性能多核处理器已开始在该系统中占主导地位，嵌入式操作系统已从简单走向智能化，与互联网、云计算结合日益密切。因此，嵌入式系统应用日益广泛。

1.1 轮询系统（Polling Systems）

嵌入式系统发展初期，嵌入式软件的开发是基于汇编语言和 C 语言直接编程，不需要操作系统的支持，这样的系统也称为裸板嵌入式系统。

用过 8051 单片机的人都知道，8051 单片机的程序从开始到结束基本上都是顺序的，最后必定有一个类似于 while 的死循环。这种方式必须不停地去轮询条件来查询要做什么事，因此这样的嵌入式系统被称为轮询系统，该方式虽然实现了宏观上执行多个事物的功能，但有以下几个明显的缺点。

（1）轮询系统是一种顺序执行的系统，事物执行的顺序必须最开始就确定，缺乏动态性，减少了系统的灵活性，也增加系统设计的复杂度。

（2）系统运行过程中无法接收和响应外部请求，无法处理紧急事情。

（3）事物之间的耦合性太大，这主要是因为事物不可剥夺的原因，正因为不可剥夺，导致一个事物的任何错误都会使其他的任务的长久等待或错误。

1.2 前后台系统（Foreground Background Systems）

针对轮询系统的不足，工程师们提出了前后台系统：后台系统与轮询系统一样也是顺序执行的，只有一个 main 程序，程序功能的实现是依靠死循环来实现；但前台引入了中断机制，

能处理外来请求。对于实时性要求比较高的事物，可以交给中断服务程序（ISR）进行处理，因为中断处理速度快，而对于非实时性的事物，可以交给后台顺序执行。

虽然前后台系统能对实时事物做出快速响应，增加了系统的动态性和灵活性，但存在如下不足：

（1）事物不可剥夺，比如说某个事物在执行的过程中，其他事物不可能执行，也就是说事物没有优先级，这与实际的情况有很大出入，实际系统中事物是有优先级的，有些任务来了很紧急，必须先执行。

（2）事物不可阻塞，也就是说事物没有暂停这一功能来阻塞自己，因为这种方式暂停当前的事物就意味着整个系统都暂停了，同时事物必须返回，因为只有这样其他的事物才能有机会执行。

1.3　嵌入式操作系统

由于前后台系统并不能很好地解决多任务并发执行的问题，尤其当系统要处理的事务和要响应的外部中断比较多时，系统的维护性就很差。更关键的是，随着嵌入式系统复杂性的增加，系统中需要管理的资源越来越多，如存储器、外设、网络协议栈、多任务、多处理器等。这时，仅用轮询系统或前后台系统实现的嵌入式系统已经很难满足用户对功能和性能的要求。因此，工程师们设计了嵌入式操作系统，以解决事务的不可剥夺、不可阻塞性，实现多任务的并发执行。由于嵌入式操作系统及其应用软件往往被嵌入到特定的控制设备或者仪器中，用于实时地响应并处理外部事件，所以嵌入式操作系统有时又称为实时操作系统（Real-time Operation System，RTOS）；另一方面，由于 RTOS 也往往存在于嵌入式系统中。因此，本书约定：为了描述方便，下文提到的 RTOS 代表的就是实时操作系统或嵌入式操作系统，以下嵌入式实时操作系统。

1.3.1　简单内核

RTOS 可简单认为是功能强大的主控程序，系统复位后首先执行；它负责在硬件基础上为应用软件建立一个功能强大的运行环境，用户的应用程序都建立在 RTOS 之上。在这个意义上，RTOS 的作用是为用户提供一台等价的扩展计算机，它比底层硬件更容易编程。一个简单的 RTOS 需要至少实现如下功能：

（1）任务调度。有了操作系统，多个任务就能并发执行，但是系统中的 CPU 资源是有限的（如单核环境下只有一个 CPU 核），于是，需要特定的调度策略来决定哪个任务先执行？哪个任务后执行？哪个任务执行多长时间等？而要实现特定的调度策略、支持多任务并发执行，还必须有任务切换机制的支持。当前各种操作系统的任务切换本质上是为了解决任务的不可剥夺和不可抢占性，任务切换可分为以下两种。

① 被动切换：也就是被剥夺（对应上面的第一条），这主要是因为优先级高的任务来了，或者当前任务的执行时间完了。

② 主动切换：也就是当前任务调用相关函数主动放弃 CPU，这对应上面的第二点，阻塞自己，让别人去使用 CPU。

（2）任务协调机制。要实现特定调度策略，除了任务切换机制外，还需任务协调机制的

支持，即任务的互斥、同步、通信机制等。这个就是大家通常说的互斥量、信号量、邮箱等。互斥量分为普通互斥量、优先级继承的互斥量（解决了优先级反转）、天花板协议的互斥量（解决了死锁问题）。

（3）内存管理机制。系统中多个任务并发执行，所有任务的执行代码和所需数据都是存储在内存中的，那各个任务及相关数据如何被分配到内存中的？这就需要 RTOS 提供内存管理机制。当然对于不同的应用需求，内存管理机制会不一样，对于 PC 等桌面应用，内存管理着重考虑的是如何有效利用内存空间，实时性不是特别重要；而对于嵌入式实时应用，内存管理的重点是内存分配和释放时间的确定性，因此 RTOS 中，内存管理的动态性就少许多。

RTOS 的出现是随着嵌入式系统的发展的必然结果，RTOS 的出现极大地推动了嵌入式系统的发展及应用，而嵌入式系统的发展，又促进了 RTOS 的不断完善和演化。据统计，到目前为止，世界各国数十家公司已成功推出 200 多种 RTOS，其中包括 WindRiver System 公司的 VxWorks、pSOS+，Mentor Graphics 公司的 VRTX，Microsoft 公司支持 Win32 API 编程接口的 Windows 8，Symbian 公司的 Symbian OS，苹果公司的 IOS，Enea 公司的 OSE，Microware 公司的 OS-9，3COM 公司的 Palm OS，国产的 DeltaOS，以及多种多样的嵌入式 Linux 等。

1.3.2 RTOS 结构

现有 RTOS 所采用的体系结构主要包括整体结构、层次结构、微内核结构和构件化结构等。

1. 整体结构

这是最早出现并一直使用至今的 RTOS 体系结构。这种 RTOS 是一个整块，内部分为若干模块，模块之间直接相互调用，不分层次，形成网状调用模式。其工作模式分为系统模式和用户模式两类：用户模式下系统空间受到保护，并且有些操作受限制；而系统模式下可访问任何空间，可执行任何操作。

从某种角度上讲，当一个拥有强大功能的 RTOS 内核被完整地应用在嵌入式环境下，就会给嵌入式软件的开发提供非常完整的平台，最常见的应用是嵌入式 Linux 和 Windows CE。对于简单的小系统而言，整体结构有几乎最高的系统效率和实时性保障。

但是，若将这种结构用于较复杂的嵌入式系统，需要大量昂贵的硬件资源；而由于内核的复杂性，使得系统的运行变得不可预测和不可靠。此外，随着嵌入式软件规模的扩大，由于模块间依赖严重，整体结构的 RTOS 在可剪裁性、可扩展性、可移植性、可重用性、可维护性等方面的缺陷越来越明显，严重制约了其应用。

2. 层次结构

层次结构也是许多流行 RTOS 选择的体系结构。这种结构中，每一层对其上层而言好像是一个虚拟计算机（Virtual Machine），下层为上层提供服务，上层使用下层提供的服务。层与层之间定义良好的接口，上下层之间通过接口进行交互与通信，每层划分为一个或多个模块（又称为组件）。在实际应用中可根据需要配置个性化的 RTOS。

内核位于 RTOS 的最底层，在某些简单的实时系统中，内核是唯一的层。内核最基本的工作是任务切换，此外，还提供了任务管理、定时器管理、中断管理、资源管理、消息管理、队列管理、信号管理等功能。RTOS 的其他组件包括内存管理、I/O 设备管理、嵌入式文件系统、嵌入式网络协议栈、嵌入式 GUI 等。

流行 RTOS 中，VxWorks、DeltaOS 等在整体上都是这种模型的范例。

3. 微内核结构

微内核结构也可称为客户机/服务器（Client/Server，C/S）结构，是目前的主流结构之一，最具有代表性的范例是 QNX。

按最初的定义，微内核中只提供几种基本服务：任务调度、任务间通信、底层的网络通信和中断处理接口，以及实时时钟。整个内核非常小（可能只有几十 KB），内核任务在独立的地址空间运行，速度极快。

传统操作系统提供的其他服务，如存储管理、文件管理、中断处理、网络通信等，都以内核上的协作任务的形式出现。每个协作任务可以看成是一个功能服务器。用户应用任务（客户任务）执行中若需要得到某种服务，则透过内核向服务器任务发出申请，由服务器任务完成相应的处理并将结果返还给客户任务（称为应答）。

随着时间的推移，微内核结构的定义已经有了显著变化。只要保持 C/S 结构，微内核中基本服务的个数不再受限，如加入基本存储管理。当然，微内核的大小尺度也有一定的放宽。在这种体系结构下，任务执行需要增加一定的开销（服务器与客户之间），与整体系统相比有一定的性能下降。但是，这种改变的好处也是十分明显的。

除基本内核外，C/S 结构的其他服务模块可以根据应用需求随意剪裁，十分符合 RTOS 的发展要求；C/S 结构可以更方便地扩展功能，可以更容易做到上层应用与下层系统的分离，便于系统移植，可以大大加强 C/S 结构服务模块的可重用性。

随着硬件性能的不断提高，内核处理速度在整个系统性能中的所占比例会越来越小，RTOS 的可剪裁性、可扩展性、可移植性、可重用性越来越重要，再加上微内核结构本身的改进，其应用面将会越来越广。

4. 构件化结构

随着构件化技术的广泛使用，如何将构件技术成功地应用到嵌入式操作系统中，受到人们越来越多的重视，这成为研究的热点之一。构件化 RTOS 内核由一组独立的构件和一个构件运行管理器构成，后者可以维护内核构件之间的协作关系。RTOS 传统的各类服务，包括任务管理、调度算法、中断管理、时钟管理、存储管理等，可以是一个构件，也可由这些相互协作的构件构成，同时为上层应用软件开发提供统一的编程接口，支持应用软件的有效开发、运行和管理。

所有的 RTOS 抽象都由可加载的构件实现，配置灵活，裁减方便，能够很好地适应各种应用领域的不同需求。作为动态构件的任务，可以自动加载运行，不需要由用户去逐一启动。构件之间具有统一标准的交互式界面，既便于用户掌握，又方便应用程序开发。

一个典型的构件化 RTOS 是 TinyOS，为无线传感器网络（Wireless Sensor Network，WSN）开发的构件化嵌入式操作系统，适用于内存资源和处理能力十分有限、电池供电的嵌入式系统。

1.3.3　多核 RTOS

随着嵌入式系统复杂度的提高，传统单核处理器及 RTOS 不能满足应用的需求。与此同时，近几年在 MIT 举行的 High Performance Embedded Computing Workshop （HPEC）以及各处理器设计/制造商纷纷推出的多核处理器，标志了多核时代的到来。可以预料，在未来较长

的一段时期内，多核计算将是计算机技术、嵌入式实时技术的一个重要发展方向。如何从传统的单核计算向多核计算过渡，成为了目前计算机及相关领域研究的热点。

操作系统作为运行在处理器上的最重要的基础软件，更成为了多核计算技术中受到普遍关注的焦点。尽管目前主流的操作系统已提供了对于桌面计算机多核处理器的支持，但是这种支持只是很浅薄的（如仅仅提供了简单的、以负载均衡为目的资源管理策略），与嵌入式实时系统对多核支持的要求相差甚远。此外，尽管一些商用嵌入式实时操作系统（如 Vxworks、QNX 等）提供了多核支持，但这种支持也是很浅薄的，并不能很好发挥多核处理器的优势和性能。因此，要让多核技术在嵌入式实时计算领域能有效应用，还有很长的路要走，这也成为当前学术界和工业界的研究热点。

1.4 从裸板开始

大家可能会认为上述介绍没什么意义，因为像这样的内容互联网上一大堆。是的，上述文字确实有些空洞，这里只是为了和大家一起梳理一下嵌入式系统软件开发及 RTOS 的演化。接下来，本书将以一款流行、易学的嵌入式开发平台 ARM（Advanced Risk Machine） Mini2440（CPU 是三星 ARM 9 系列的 ARM S3C2440）[1-2][37]为例，通过具体代码实现，介绍如何从裸板入手设计简单的轮询系统、前后台系统，以及如何一步一步在 ARM Mini2440 上编写 RTOS 内核，到如何让 RTOS 内核支持多核嵌入式处理器。本书要求读者已掌握 ARM 9 处理器的基础知识，并且对 ARM 汇编、ADS 编译器、Linux 交叉开发环境的使用有足够了解（这部分内容不属于本书的范围，但却是本书的重要基础）。

到地，该引入本书的主角 aCoral 了[1][3][4]，aCoral 是一个开源 RTOS，本书将以它为例，介绍其设计与实现。具体而言，本书将以 aCoral 的设计为主线，详细剖析：ARM Mini2440 是如何开始工作的？上电后运行的第一行代码是什么？各个外部设备如何驱动？aCoral 是如何启动 Mini2440？如何管理 Mini2440 的硬件设备？什么时候开始创建第一个任务？多个任务如何并发执行？如何切换并发执行的任务？调度算法如何在 aCoral 中实现等？

1.5 aCoral

aCoral 是一款由电子科技大学实时计算实验室于 2009 年创建的开源的、支持多核（Symmetry Multiple Process，SMP）的 RTOS[1][3][4][42-45]。aCoral（A small Coral），珊瑚的特性是 aCoral 追求的目标。aCoral 具有高可配，高扩展性，可以在 www.aCoral.org 下载其源代码、文档及基于 aCoral 的应用开发实例，如 JEPG 的并行压缩、基于 aCoral 的网络收音机等。目前的 aCoral 包括五大模块：

（1）内核：由电子科技大学实时计算实验室编写。

（2）文件系统：在周立功文件系统上进行了优化而来。

（3）轻型 TCP/IP（LWIP）：由 LWIP 移植而来。

（4）GUI（TLGUI）：来自开源嵌入式 Linux 图形系统 LGUI。

（5）简单应用（网页服务器、Telent 服务、文件操作、GUI 图形、测试等）。

aCoral 支持多任务模式，其最小配置时，生成的代码为 7KB 左右，而配置文件系统，轻型 TCP/IP，GUI 后生成的代码仅有 300KB 左右。目前，aCoral 支持各种 ARM 系列处理器：

Cortex-m3、ARM7、ARM9、ARM11，以及 ARM11 MPCore 四核平台[1][6-7]。同时，为了方便没有开发板的用户体验 aCoral，其模拟版本可以在运行 Linux 的 PC 中作为应用程序运行，这种模式可以在 PC 上体验 aCoral 的所有功能，包括内核、文件系统、GUI，该模式支持单核和多核。

实时计算实验室在多年的本科生教学[《嵌入式实时操作系统（ERTOS: Real-Time Operating System）》]、研究生教学（《嵌入式系统开发》、《实时计算》、《可信计算》）、留学生教学[《嵌入式操作系统及应用（Real-Time Operating System and application）》]、海外学生教学《Embedded System and Real-Time》）中发现，学生们在学习 RTOS 时，常常会有许多疑惑，并且很难有自己动手写 RTOS 的机会，这让大家难以融会贯通《计算机组成原理》、《C 语言》、《汇编语言》、《操作系统》、《数据结构》、《嵌入式系统开发》的知识点，从而对计算机系统结构缺乏系统的、深入的理解。创建 aCoral 开源项目的目的是：希望能激发同学们自己编写操作系统的热情（写出操作系统不是目的，目的是让大家在写的过程中能真正思考和深度理解操作系统及相关知识点（编译、链接、加载等）的原理，从而提高分析问题、解决问题的能力，使计算机系统能力、工程能力上一个新台阶）。希望 aCoral 为对嵌入式实时操作系统有兴趣的同学或朋友们提供了一个较好的学习蓝本，aCoral 将应用在未来的本科教学和研究生教学中。

电子科技大学实时计算实验室在 RTOS 方面有着长期持续的研究，这里孕育了中国航空工业集团下属的科银京成技术有限公司。因此，aCoral 的另一发展思路是：多核 + 强实时，为对性能有苛刻要求（Performance-critical）的嵌入式实时系统[如计算机密集型的嵌入式实时应用（高端控制系统、超声波无损检测与处理系统、精确导航与防撞系统）、航空电子系统等]提供一体化解决方案，充分发挥多核潜能，力求系统总体性能最佳。

对于多核， aCoral 已支持同构多核（如 ARM11 MPCore 四核平台）[39-41]，对于异构多核的支持，项目组已在 AMR+DSP 构架下，实现了基本的异构通信、同步、共享内存等机制，对更高级别的支持（异构多核调度、对 GPU 的支持等）正在研究中[33-35][46]。对于强实时而言，嵌入式操作系统一般都是实时的，但是如何做到强实时是一个很棘手的问题，为强实时计算密集型应用[17]（航空电子、舰载电子等）提供可靠运行支持是 aCoral 开发的强力主线。目前 aCoral 提供了强实时内核机制（优先级位图法、优先级天花板协议、差分时间链、最短关中断时间等）。与此同时，aCoral 还提供了强实时调度策略：单核和多核的 RM 调度算法，由于多核情况下的 RM 算法的复杂性，目前只支持简单环境下多核 RM 调度， RM 调度算法在多核情况下的其他问题正在研究和解决中。此外，其他多核强实时确保策略也正在研究中。

aCoral 会像珊瑚一样成长……

1.6　本书结构

本书将介绍嵌入式系统的发展，早期嵌入式软件轮询系统的基本原理和缺陷，前后台系统的出现及不足，RTOS 的诞生及发展。本书基于三星公司 ARM9 Mini2440 开发板逐步介绍轮询系统、前后台系统到 RTOS 是如何设计和实现的。

第 1 章：主要介绍 RTOS 的发展、主流结构及 aCoral 基本情况。

第 2 章：主要介绍轮询系统的基本原理，以及一个基于 ARM9 Mini2440 开发板的简单轮询系统的设计和实现。

第 3 章：在轮询系统基础上引入中断机制，介绍前后台系统的基本原理，然后重点描述裸板环境下中断发生、响应、处理的流程；在此基础上，详细介绍一个基于 ARM Mini2440 开发板的简单前后台系统的设计和实现。

第 4 章：介绍 RTOS 相关的基本概念，为 aCoral 在 ARM Mini2440 上的设计和实现做一个理论上的铺垫。

第 5 章：介绍 aCoral 的交叉开发环境，以及如何在 Ubuntu 下搭建其交叉开发工具链。

第 6 章：从用 C 语言描述一个线程开始，一步一步阐述 aCoral 内核的设计与实现，主要内容包括创建线程、调度线程、aCoral 事务处理机制（中断与时钟管理）、内存管理机制、线程间的交互机制（通信、同步、互斥等）。

第 7 章：介绍 aCoral 是在 ARM9 Mini2440 上如何启动并开始执行的。

第 8 章：介绍 aCoral 硬件抽象层（HAL）相关代码，通过对这些代码的分析，大家可知道如何将 aCoral 从一个嵌入式平台移植到另一个嵌入式平台上，例如，从 ARM9 s3c2440 处理器移植到 ARM s3c2410 处理器。

第 9 章：基于 aCoral 工程文件，介绍如何在宿主机上编译 aCoral，如何在 ARM Mini2440 上运行 aCoral。

第 10 章：简单介绍 RTOS 的实时调度策略，实时系统运行过程中遇到的经典问题以及相应的解决办法。

第 11 章：以四核嵌入式处理器 ARM11 MPCore 为例，从操作系统引导和线程调度等方面介绍 aCoral 是如何支持多核处理器的。

习题

1. 简述嵌入式系统的发展历程。
2. 轮询系统的特点是什么？适用于哪些嵌入式应用？
3. 前后台系统有什么优缺点？
4. 嵌入式操作系统包括哪些结构？对比各种结构的特点。
5. 嵌入式操作系统内核的协调机制有哪些？