第 3 章

嵌入式操作系统

首先，对嵌入式操作系统的基本概念和术语做一个详细的解释，以便后面讲述时查询并使用。

3.1 嵌入式操作系统的定义

嵌入式操作系统(Embedded Operation System，EOS)从连接功能上看，是硬件与上层应用软件的连接软件。其主要责任是为上层软件提供脱离硬件运行的支撑平台，同时为与下层硬件沟通提供访问程序接口；从功能上看，嵌入式操作系统要能够管理好系统的各种资源和硬件，负责嵌入式系统的全部软、硬件资源的分配、任务调度，控制、协调并发活动，使系统高效可用。一般地，它将体现其所在系统的特征，能够通过装卸某些模块来达到系统所要求的功能。

嵌入式操作系统是随着嵌入式设备的开发而产生的。这些嵌入式设备就是通过微处理器来控制，如微波炉、电视机或者手机等电子设备。因此嵌入式操作系统具有一些实时系统的特性，另外还有如内存、能量的限制等。

为此，设计嵌入式操作系统是一个“量体裁衣”的过程。首先要熟悉所要运行的软、硬件环境；其次，对于具体的应用，要明确其微处理器是什么，是否要跨平台、跨语言、跨应用，是否要有网络支持和文件支持；最后才能确定设计目标，更好地进行功能划分。

第 5 集
视频讲解

3.2 嵌入式操作系统的体系结构

设计一个嵌入式操作系统，必须先定义好其体系结构，可分为整体型、层次型、微内核和客户-服务器体系结构。同时，嵌入式操作系统体系结构也体现了一个操作系统的整体脉络，了解了这个脉络，对整个操作系统的理解和使用也就有了着手点。

3.2.1 整体型体系结构

整体型体系结构其实就是无结构。任何过程都可以随意调用其他过程，因此没有信息

隐藏的概念。这类结构中，过程代码接口需要精心设计，以方便其他过程调用。

在这种类型的嵌入式操作系统中，如图 3.1 所示，把所有处理分成 3 个层次，使设计显得比较有条理。其与层次型体系结构的不同在于：上下层处理都不隐藏信息，都可以互相调用。例如，最早的 DOS(Disk Operating System，磁盘操作系统)就是一个整体型体系结构，模块之间可以相互调用。

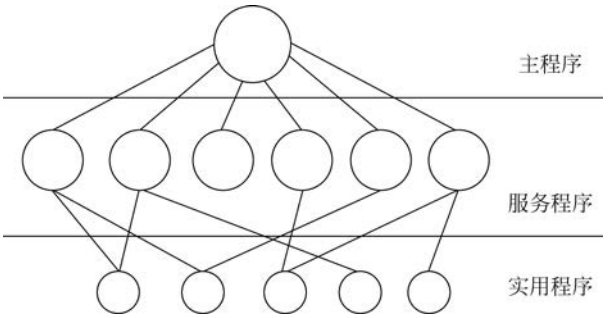


图 3.1　嵌入式操作系统整体型体系结构

3.2.2　层次型体系结构

层次型体系结构是最简单和自然的划分方法，也就是把整个设计按照离硬件的远近层次来划分。层次型体系结构设计一般来说每层都会对相邻层隐藏一些信息，非相邻层完全不能够互相调用。

用这种结构设计一个新系统时必须小心地安排各层的功能。最底层是最靠近硬件的，主要目的是对上层隐藏硬件细节。紧挨着的上一层就可以设计成中断处理、上下文切换、内存管理单元等。再往上的层次就跟硬件完全无关了。例如第 3 层用于线程、线程同步和线程调度，可以设计成多任务、多线程管理，第 4 层可以设计一些任务之间的通信等。这样的设计可以简化 I/O 处理的结构。当 I/O 中断产生后，可以利用进程间通信，由调度程序关掉使用该中断需要的资源的线程，使中断处理程序处于获得所有资源的状态。UNIX 就采用这种方式，但是 UNIX、Linux 和 Windows 都没有使用这样主动处理中断的方式。其实操作系统中最复杂的部分就是 I/O 的处理，任何可以简化操作的技术都是可取的。

图 3.2　嵌入式操作系统层次型体系结构

例如，图 3.2 中内存管理放在文件系统下层是为了能够完成文件读/写时高速缓存的换入、换出操作。

又如，Linux 就是一个层次型体系结构，如图 3.3 所示。

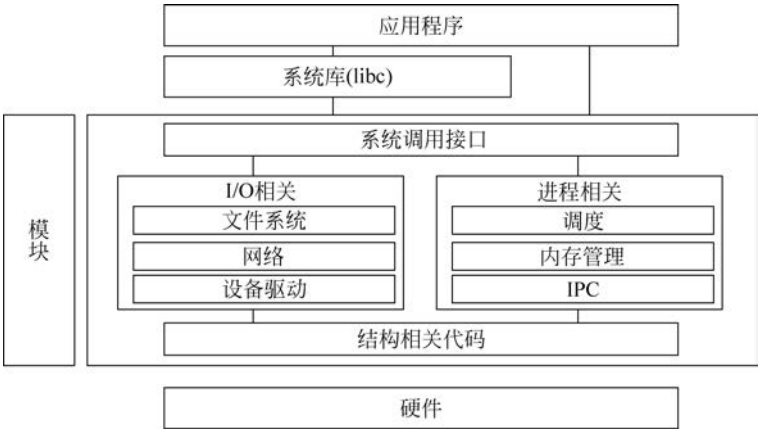


图 3.3 Linux 的层次型体系结构

3.2.3 微内核体系结构

与层次型体系结构划分相反的是，一些人认为没有必要由嵌入式操作系统完成那么多功能，必须把一部分开发的自由交给编程者，使他们可以根据自己的需要为嵌入式操作系统做自己的功能扩展。为此，他们主张采用微内核体系结构设计嵌入式操作系统，也就是只提供一个基本的嵌入式操作系统内核，完成如硬件的屏蔽、任务调度等，其余的如网络支持或者文件支持设计成可选模块，交给编程者来决定使用或者做拓展。嵌入式操作系统微内核体系结构如图 3.4 所示。

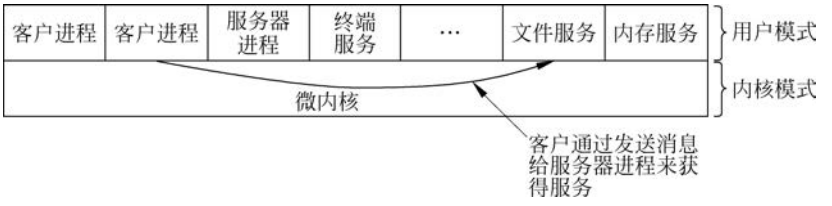


图 3.4 嵌入式操作系统微内核体系结构

3.2.4 客户-服务器体系结构

客户-服务器体系结构是针对网络环境的，在不同功能的计算机上装不同的嵌入式操作系统。在客户端用轻负载的嵌入式操作系统，一般地，其功能就是收集数据；在服务器端则装比较大的操作系统，支持更复杂的任务和更复杂的硬件。从图 3.5 可以看出，客户-服务器体系结构的嵌入式操作系统同时也是具有微内核的嵌入式操作系统，支持不同的应用程序，内核部分主要完成网络通信功能。

如图 3.6 所示是 QNX4.25 的体系结构，这就是一种客户-服务器体系结构。2011 年年底，RIM 公司整合黑莓与 QNX 公司发布 BBX 系统，也说明了这种体系结构是具有优势的。

QNX 具备一个很小的内核，同时也是微内核体系结构的操作系统。QNX 的内核一般只有几万字节，整个操作系统可根据需要定制模块。不同的客户应用可以选择不同的模块，但是都必须选择微内核部分。服务器包含微内核和更多的模块部分。

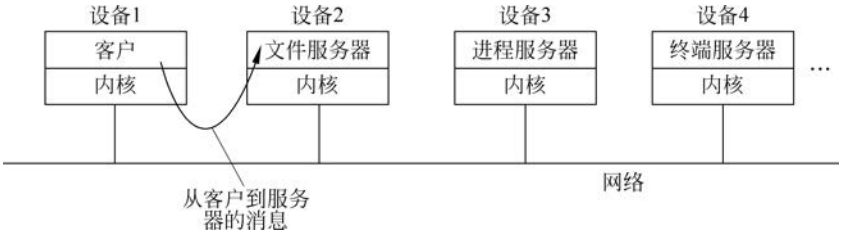


图 3.5　嵌入式操作系统客户-服务器体系结构

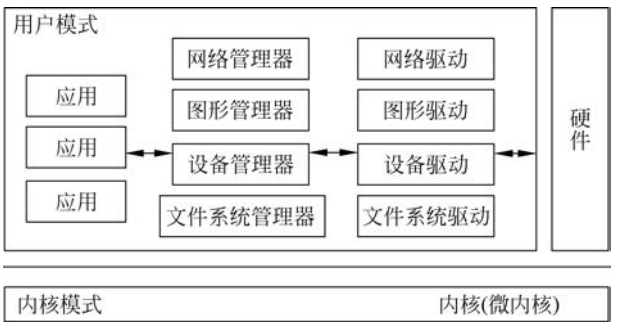


图 3.6　QNX4.25 的体系结构

当然，以上几种体系结构是可以混合运用的。

3.3　嵌入式操作系统的组成要素及概念

嵌入式操作系统虽然各有不同，但是基本来讲，都是针对各类硬件资源管理的，包括针对分享 MCU(Microcontroller Unit，微控制单元)的任务管理、消息机制、同步机制等，针对接口硬件的中断处理，针对内存的内存管理、文件管理，以及针对网络的网络支持、网络管理等。在深入学习嵌入式操作系统之前，先介绍嵌入式操作系统的组成要素及其概念，如任务、实时、多任务、任务状态及转化、内核、调度等。

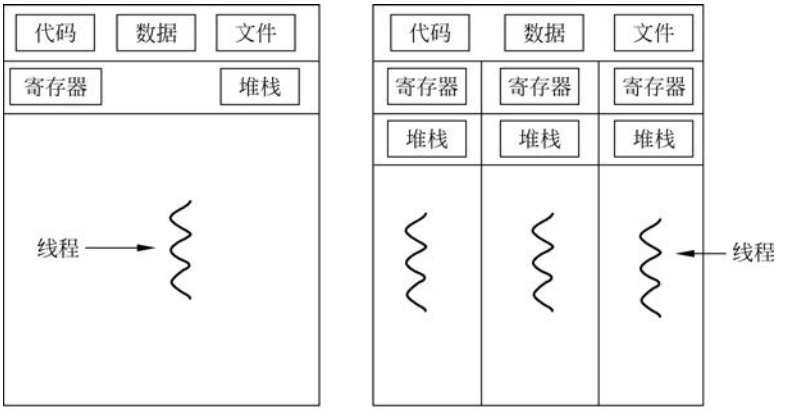
1. 任务

任务是一个抽象的概念，进程和线程都只是任务的一个特例。简单而言，嵌入式操作系统中的任务是一段无限循环的代码，在这段代码执行的过程中有相应的堆栈、内存的分配。

每种特定的嵌入式操作系统都有自己的描述单位。例如，Windows CE 中以进程为基本单位来描述资源，每个进程一旦运行，操作系统要为其开辟相应的内存空间，供其进行临时数据存储等操作。线程则被 MCU 实际调度，是调度的实体。一个进程创建之后，同时将

创建一个主线程，可以在主线程中创建该进程的其他线程。进程可以被视为线程的容器。一个线程默认的栈大小为 64KB，也可以在创建线程时自定义栈的大小。同一个进程中，一个线程分配的内存可以被其他线程所访问。不同进程中的线程如要互相访问，则需要通过进程间通信来处理。在 Symbian 操作系统中，每个进程都有一个或多个线程。线程是执行的基本单位。Linux 中线程和进程则更加含混，都使用任务这个结构来描述。

比较线程和进程，总地来说，进程的描述粒度较大，涉及内存空间的划分，不涉及具体微处理器的寄存器等，离硬件的距离比线程远。线程在运行时涉及具体寄存器等保存和上下文环境切换，由微处理器进行调度，与微处理器的资源联系紧密。图 3.7(a)是单线程进程的内存运行模式，图 3.7(b)是多线程进程的内存运行模式。在图 3.7 中，每个线程都拥有自己的寄存器和堆栈，而每个进程只拥有自己的代码、数据和文件。



(a) 单线程进程的内存运行模式　　(b) 多线程进程的内存运行模式

图 3.7　单线程进程和多线程进程

2. 实时

实时性是一般嵌入式操作系统有别于其他通用操作系统的一个特点。这里的实时性是指系统的正确性不仅取决于逻辑运算结果的正确，而且取决于结果传递的时间必须在时限要求内。能支持这样的实时性的嵌入式操作系统称为实时嵌入式操作系统，换句话说，就是需要能为不确定发生时间的外部事件做出时间确定的反应。好的实时嵌入式操作系统必须在所有情况下运行都能获得确定的结果。这样的操作系统小到 10KB，大到 100KB。必须满足的时限称为硬实时。如果允许偶尔地打破时限，并在这种不正常情况下还能恢复到常态，即满足时限状态，则称为软实时。

3. 多任务

多任务是实现实时性的一个好方法。如果是在一个处理器上实现多任务，则是一种多任务的伪同步方式。此时，多任务分时使用处理器。在一段时间内看起来多任务同时使用处理器，也称为多任务并行。这种方式需要额外的控制，如时间的划分、当前运行任务的退出，以及选取任务进入处理器进行处理等。如果是多处理器，在同一时刻，多个任务可以运

行在多个处理器上，但是需要额外的控制机制来保证任务运行整体的确定性。多处理器上的多任务目前仍在研究阶段。

对于单处理器的这种多任务同步的实时性，需要考虑任务从微处理器换入、换出的时间。而进程的概念比较大，这种换入、换出所涉及的内容太多，无法满足实时的要求。为此，在多任务的系统中常常使用线程作为基本单元。这也就是现在对于实时系统只提及任务或者线程的原因。

4. 任务状态及转化

嵌入式操作系统中任何任务一般都处于以下 3 种状态(见图 3.8)。

运行状态(Running)：此任务正占有微处理器并正在获得执行。在单处理器系统中，某个时刻仅有一个任务在运行。

就绪状态(Ready)：此任务已获得所有可拥有的微处理器来运行的条件，而此时某个其他任务正占有微处理器，此任务必须等着微处理器空闲。

阻塞状态(Blocked)：此任务还不具备一些条件，此时就算将微处理器分给该任务，在此状态下也无法运行。在此状态下的任务意味着在等待一些外部事件，如在等待网络传来数据，而数据还没有到达的情况。

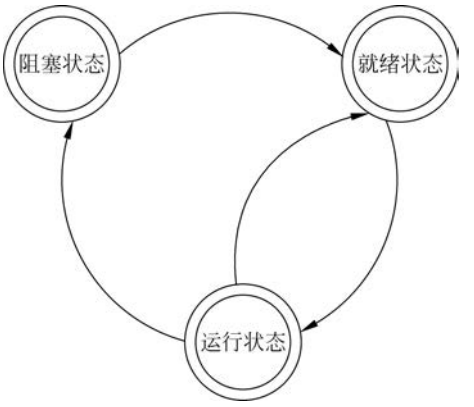


图 3.8　任务状态图

5. 内核

嵌入式操作系统中有一部分功能是所有操作系统都有并构成操作系统的核心，纵使如何裁剪系统，也无法去掉的部分，称为内核。一般内核可以按功能分为几块：任务调度、消息处理、内存管理、中断处理和时间管理。

6. 调度

调度是嵌入式操作系统的灵魂。所谓调度包括两个大概念和一个小概念。第一个大概念是指如何分配微处理器给就绪状态的任务的换入策略；第二个大概念是指正在占有微处理器的任务的换出策略。小概念是指当有正在执行的任务被中断换出时，中断结束后是回到刚才的任务还是重新选择任务，也就是抢占还是不抢占的调度问题。

在就绪状态等待微处理器的任务不止一个，到底哪个能占有微处理器而进入处理状态，是需要策略的。最简单的换入策略，如先进先出，哪个先到达就绪状态，就先分微处理器给哪个。但是在嵌入式操作系统中一般采用基于优先级分配策略。所谓基于优先级分配策略，就是每个任务根据重要程度分配其优先级别，哪个就绪任务的优先级高，哪个任务就可以占有微处理器。

正在执行的任务何时换出？任务执行完自然会换出。但是一般任务都是一段循环代码，执行起来会进入反复操作，为此给每个任务分配的时间，又称为分配的时间片。该任务

的处理时间用完了，也就是该换出的时候了。另外，按照优先级的思路，如果有比正在运行的任务优先级更高的任务到了就绪状态，正在运行的任务也该让出微处理器，此时也需要换出。

特殊的是中断，中断是来自硬件的信号，必须及时处理。一般中断到达后，当前在运行的任务需要换出，让出微处理器给中断使用。中断处理后返回刚才被中断的任务称为非抢占式调度。中断处理后所有任务重新按换入策略选择，称为抢占式调度。

3.4 嵌入式操作系统编写的要求

嵌入式操作系统对程序编写有特殊的要求。

第一个要求：程序执行时间可知。在写嵌入式操作系统时，查询不可以采用循环嵌套。比如要找一个等于 3 的数，不可以采用循环方式来写，那样，如果 3 存放在第 2 个数组中的前面，则找 2 次就能找到，循环执行 2 次；而如果 3 存放在第 100 个数组中，则需要循环执行 100 次。2 次循环与 100 次循环程序的执行时间长短变化太大，将导致操作系统执行时间未知范围过大。

因此需要精心构造算法，使得查找快速，完成查询基本定时长。

第二个要求：建立索引。为了能够定时长并且能在大量数据的情况下快速查找，需要大数思维。大数思维，就是指在如图书馆那么多的图书时，为了快速查询到一本书，需要建立索引。操作系统会对内存、任务、消息等进行管理。运行过程必须有索引，方便快速找到所要的内存块、任务指针等。

第三个要求：程序模块尽量抽象重用。这个要求是为了能够尽量避免错误，也为了能够使得程序模块化更好。

第四个要求：空间重用，大小可知。采用空间可回收机制。本身就是操作系统，没有更底层的程序会来打扫空间，所以对于空间的使用，必须自己用、自己清理。如前面讲过的存储器，存储地址空间是连续的。如果操作系统不能够自己限制自己的执行空间大小，将可能造成程序越界的情况，而使得系统陷入瘫痪。所以一般操作系统的使用空间大小是可以确定的。比如在配置系统时就已经可以限制任务的多少，从而决定要使用多少地址的空间开销。整个系统在执行过程会时刻保持边用边回收的策略。

第五个要求：灵活性。要为高级程序员开放内核，便于修改内核。嵌入式操作系统将会在程序中采用一些面向对象可重构的思想，让程序既可以完整运行，又可以在需要的时候改变重构内核来运行。

习题

1. 嵌入式操作系统的体系结构是什么？
2. 嵌入式操作系统的定义是什么？
3. 任务有几个状态？是如何相互转化的？