开源嵌入式实时操作系统

RT-Thread发展历程

文 / 熊谱翔

在很多人看来，开发操作系统是一个庞大而艰巨的工程，涉及多任务调度、内存管理、文件系统、设备驱动等多种功能模块。目前最流行的开源系统GNU/Linux，仅内核源码已超过1000万行。本文中，RT-Thread的作者将介绍这款国产嵌入式操作系统的诞生与发展。

RT-Thread是一套国产开源嵌入式实时操作系统。第一版发布于2006年，最初仅包含基本的内核和几个组件，经过6年的发展，现已成为一套完善的、有众多企业应用的嵌入式实时操作系统平台。当前RT-Thread系统版本为1.1.0，主要涵盖下面一些部分（系统结构如图1所示）。

- Kernel，稳定且高度可裁剪的实时内核。
- Finsh，一个支持C语言表达式的命令行工具、调试工具。
- DFS设备虚拟文件系统，支持FAT16/32、NFS、JFFS2、YAFFS2、UFFS等多种文件系统。
- 轻型TCP/IP协议栈，采用专为RT-Thread优化的LwIP。
- 设备驱动框架（USB Host/Device stack、SPI/I²C BUS等）。
- 多窗口多线程图形界面组件RT-Thread GUI。
- 应用模块（App Module）支持。
- Lua脚本支持。

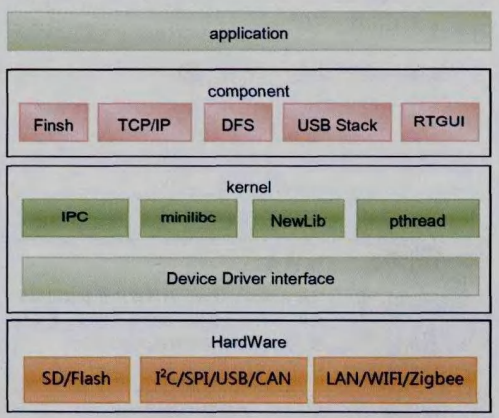

图1 RT-Thread系统框架图

前世今生

RT-Thread的诞生很偶然，2005年底我与朋友开发一个简单手持设备项目，当时低成本芯片主要是ARM7TDMI，用于存放程序的Flash和变量的RAM都内置在芯片中，系统资源非常紧张，只有16KB内存，比DOS时代PC机资源都稀缺。

当时也存在一些嵌入式实时操作系统，商业的如VxWorks、μC/OS-II；开源的则有eCos、RTEMS以及μClinux等。对于一家小型企业来说，商用RTOS费用高昂，再加上我已习惯于UNIX的编程风格，对μC/OS-II编程风格实在无爱，而开源的嵌入式操作系统却不能够达到体积小的指标，例如μClinux动辄占用上兆内存。

在这种情况下，我尝试着自己编写一个小型的嵌入式实时操作系统，在花费了大量的业余时间之后，2006年初有了第一个内核版本（0.1版）。因为RTOS中任务更类似于通用操作系统中的线程，并

且这个系统支持基于优先级的抢占式任务调度算法，调度器的时间复杂度是O(1)，所以把它命名为RT-Thread（Real-time-Thread），即实时线程。我在年中时实现了完整的线程间通信机制，包括信号量、互斥量、邮箱、消息队列等，并发布了0.2版。后来朋友的项目不幸夭折，但RT-Thread却因为开源而保留下来，算是不幸中的万幸。

在创建RT-Thread项目之初，我纯粹以一个技术痴迷者去做这件事，希望在技术上达到一定高度，实现基本的功能、支持多种体系结构，渴望它能够成为深度嵌入式系统领域的Linux，但实现这个愿景并不容易。RT-Thread诞生后的两三年是项目最艰难的时期。仅有一个内核，显然会淹没在全球多达数百种操作系统的海洋中，一个泡都不会冒，因此在这段时间，我把它移植到了不同的芯片平台上，包括ARM（s3c4510、AT91SAM7S64等芯片）、ColdFire、x86等体系架构，统一使用GCC编译器。当时我的本职工作是3G协议栈研发，白天翻看着多达上千页的3GPP 25.331 RRC协议，晚上沉迷在自己的嵌入式实时操作系统世界中（这是否也是任务切换呢）。这段时间真的非常艰难吗？或许不是，但肯定是苦中做乐。这个时期发布了RT-Thread 0.2.x等不同的版本以支持不同平台。在那段苦闷的日子里，shaolin同学出现了，他完成了RT-Thread for x86的移植，当时他还在电子科技大学读书，最后毕业工作时留在了上海，相同的城市、相同的爱好，无疑大大增强了我坚持下去的信心。

到了2008年，ARM Cortex-M3在嵌入式市场上掀起一场变革。而RT-Thread对Cortex-M3的支持则源于一位网友不经意间一篇希望支持Cortex-M3的帖子。我们采纳了这个建议，并于2009年初发布了RT-Thread for STM32的测试版本。这时aozima出现了，他从LPC2148入手，后续逐渐进入到ARM Cortex-M3领域，从自己接些小项目做起，到现在成为RT-Thread for STM32的分支维护人。当RT-Thread for ARM Cortex-M3面世后，甚至有网友戏言，RT-Thread+CM3是绝配，因为Cortex-M3系列芯片通常内置了大容量Flash和RAM，而RT-Thread提供了开源、免费的实时操作系统，包括实时内核、命令行、文件系统及TCP/IP协议栈等，不仅功能丰富，而且资源占用极低。

面向企业

随着RT-Thread逐渐成熟，2010年它开始为国内一些企业所认识和了解。他们先尝试着小规模试用，当发现RT-Thread能够满足需求后，开始逐渐应用到产品上。

RT-Thread默认许可证协议是GPLv2，对于深度嵌入式系统，应用程序与操作系统内核通常链接在一起，因此按照GPLv2的条款，应用程序也需要开源。为了在商业应用上更宽松，RT-Thread选择的是GPLv2和商业许可证双重许可证方式，其商业许可非常宽松，将RT-Thread应用于产品时，只需要在产品说明书上提及“基于RT-Thread系统”或“Powered by RT-Thread”，即可免费获得商业使用许可。但第一份申请商业许可的企业出于商业保密的顾虑，却向我们申请购买了纯粹的商业许可。笔者当时想着，钱虽然不多却可以支持RT-Thread网站运营很长一段时间，真是无心插柳柳成荫！

后续在与这家企业一年多接触的过程中我们也了解到另外一种情况，企业尝试使用之后，觉得RT-Thread稳定性挺好，也很有活力，但依然担心如果使用过程中出了难以解决的问题怎么办，以后这个开源项目停止维护又怎么办？如果因为一个开源项目缺乏必要的后备支持，从而导致它不能很好地进入产品、工程领域岂不非常可惜。笔者越发觉得有必要建立一个后备技术支持团队，为企业应用提供技术支持。基于这样的考虑，2011年初我们依托RT-Thread核心开发人员，建立了一支包括数名全职技术工作人员的支持团队，为RT-Thread企业应用保驾护航。

例如一家做继电保护的企业，在使用默认版本时，达不到他们要求的中断响应时间，在我们支持服务下，对他们使用的芯片进行针对性优化，使得中断响应时间由最长100μs缩短到最长仅0.67μs。又例如一家做用电信息采集的企业，他们需要在有限内存资源的ARM Cortex-M3上使用NandFlash作为FAT文件系统的存储介质，而通常使用在Linux上的YAFFS文件系统对于128MB

NandFlash内存占用达数百KB，对于一个只有几十KB的Cortex-M3来说基本不可能。我们为其设计了贴身的带日志功能的NandFlash转换层，直接使用FAT文件系统并且上层API接口都不改变，实现了文件系统永不被破坏的目标。

我们也与一些半导体厂商形成了良好的合作伙伴关系，如ARM公司、富士通半导体、恩智浦半导体等，为它们的芯片产品提供上层系统软件支持，并在2011年底荣获了龙芯第一届开源大赛特别奖、中日韩开源大赛优胜奖等开源国际奖项。

开源社区

开源项目的发展离不开社区的支持，因为社区为开源项目提供了发展的源动力，用户可以在社区共同讨论使用心得，也是用户反馈、讨论解决办法的最佳去处。并且用户在深入了解后，非常可能转换成开源项目的开发人员，而这种开发人员往往都是那种有实际需求的人，更了解实践应用的真实场景。

但在国内，原创开源项目难免受到用户圈狭小的限制，如何培养好社区是一个值得深思的问题。RT-Thread社区初期也遇到了类似的困难，特别是当它仅有一个实时操作系统内核、功能单一时，这种现象更为严重，相应的开发人员也少。

在RT-Thread支持ARM Cortex-M3芯片后，从玩的心态出发（Just for fun），并在aozima、54et、gzhuli等网友的帮助下，以社区的方式我们制作了第一套开源硬件平台——STM32Radio网络收音机。在这个硬件平台上，我们采用开源的方式实现了自己感兴趣的事情：在意法半导体的STM32微控制器上，通过网络获取豆瓣电台并播放。参与的人员既了解了整个硬件的制作过程，也在上面使用RT-Thread内核，及其外围文件系统、TCP/IP协议栈、GUI等组件构成一套完整的软硬件平台，并深入学习了MP3软解压音频播放过程、网络编程技术，以及RT-Thread开发知识等。STM32Radio成为了当年ARM Cortex-M3上最热门的DIY作品，我们前后发行了近200套，每次STM32Radio还未面世就被预定一空，并在意法半导体全国技术巡展会中作为奖品赠送给技术研讨会的与会人员。

STM32Radio开源硬件平台让我们意识到社区的重要性，它可以让围观的技术爱好者参与进来，一起进行DIY，在这个过程中他们不仅学习到了实时操作系统的知识、如何应用，他们的反馈也让项目开发组了解到RT-Thread的不足之处，并促使我们不断改善。一些爱好者为RT-Thread源码提供补丁修补Bug，并因此成RT-Thread开源项目的开发人员。

随着参与的人越来越多，如何管理RT-Thread这个开源项目，使它顺利发展变得越来越重要，特别是RT-Thread是以追求稳定性为第一要素的实时操作系统。这些包括RT-Thread本身的代码开发推进制度，也包括RT-Thread的代码管理。

RT-Thread的开发维护采用的是维护人制度。在开发维护上，我们把整个系统按照一定方式进行分类，整个系统变成了下面几个部分。

- 内核，实时操作系统内核核心，包括多任务管理、调度、多任务间的通信、内存管理等功能。
- 组件，在实时操作系统内核外围完成一定功能的模块，例如Shell、文件系统、TCP/IP协议栈等。它们或由开发人员自行开发，或从其他开源软件平台移植过来，它们和RT-Thread完美地融合在一起，形成一个功能整体。
- 分支，针对不同的芯片平台，提供相应的底层支撑（中断管理、启动代码、任务切换汇编代码）和驱动程序。

基于这样的划分，每个功能模块都具备相应的维护人。维护人负责管理对应的模块，评审补丁提交者代码质量情况，以决定是否能够进入到开发主干中。同时项目安排了单独的发布协调人，当进行发布时，负责查看各个发布分支、发布组件的情况，并与维护人进行协调。

RT-Thread的大版本发布周期大致是一年。在进行大版本开发前，我们会从社区中征集出这个版本包含的功能集，然后进入开发阶段。在发布周期这段时间内会发布一系列的测试版本，例如先期的Alpha技术预览版本以体现下一代RT-Thread的技术进展情况；Beta版供社区进行测试反馈；RC版本作为发布前的候选版本，并冻结相应的新功

能，仅进行后续的缺陷修正；Final版本发布意味着可用于生产的版本正式推出。

挑战与机遇

开源领域，竞争尤为残酷，不仅需要提供好的技术，更为重要的是构建良性的生态环境，RT-Thread社区面临着种种挑战。

- 小型设备使用得更为广泛，这类设备普遍具有低资源、低成本的特点。
- 小型设备不再孤独，物联网提上了日程，设备需要沟通起来，低功耗也一再被关注。
- 如何营造好社区依然是一道难题，同时还有生态圈，相应的角色有哪些。

开源硬件异军突起，代表着创新与变革。基于开源硬件的DIY参与者可以抛开工作的顾虑和烦恼，体验单纯的玩带来的轻松愉悦，一些年长的开发者可以与子女一起动手，享受天伦之乐。RT-Thread遵循开源、技术分享的准则，有理由把这些DIY带给大家，带给社区，这也是我们目前构造的RT-Thread ART（一套兼容Arduino软、硬件接口的易编程开发板）计划的目标，通过ART这个硬件载体把RT-Thread的技术带给大家。

RT-Thread 1.1.0正式版将于2012年底发布，下一版本（1.2.0系列）也逐渐提上日程。RT-Thread将围绕着这些关键问题展开深入的探索，请期待下一节的登场。

熊谱翔

1998年接触Linux文化，深受开源思想影响。毕业后工作在嵌入式实时系统研发一线，参与多个大型嵌入式产品项目研发。2006年从零开始创建RT-Thread开源实时操作系统。

责任编辑：卢鹄翔（ludx@csdn.net）