LINUX

- 业界权威机构和专家强力推荐
- 多年培训、研发经验的总结

内核标准教程

Linux 2.6

华清远见嵌入式培训中心 河秦 王洪涛 编著

附源代码光盘

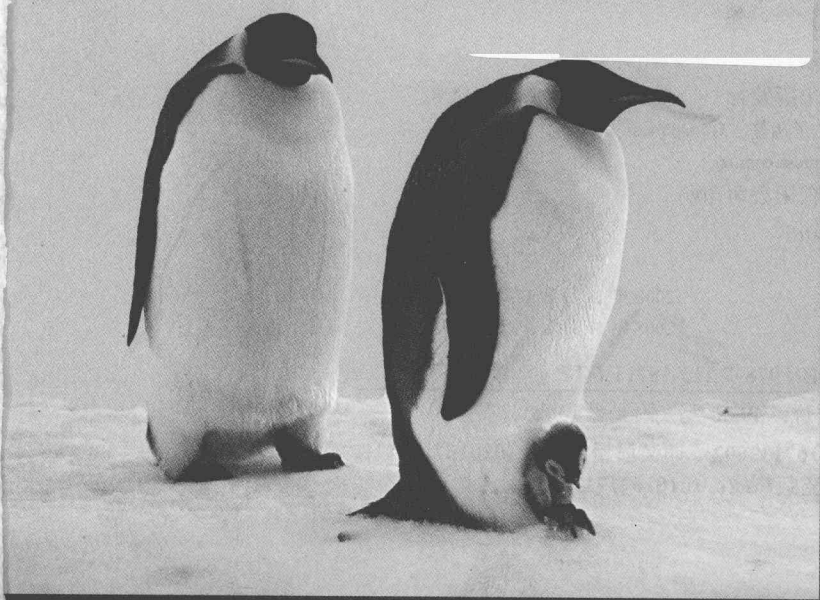
人民邮电出版社
POSTS & TELECOM PRESS

LINUX

内核标准教程

华清远见嵌入式培训中心 河秦 王洪涛 编著

Linux 2.6



人民邮电出版社
北京

图书在版编目（CIP）数据

Linux 2.6内核标准教程 / 华清远见嵌入式培训中心，河秦，王洪涛编著. —北京：人民邮电出版社，2008.11
ISBN 978-7-115-18711-6

Ⅰ. L… Ⅱ. ①华…②河…③王… Ⅲ. Linux操作系统—教材 Ⅳ. TP316.89

中国版本图书馆CIP数据核字（2008）第130219号

内 容 提 要

Linux 内核是 Linux 操作系统中最核心的部分，用于实现对硬件部件的编程控制和接口操作。本书深入、系统地讲解了 Linux 内核的工作原理，对 Linux 内核的核心组件逐一进行深入讲解。

全书共 8 章，首先讲解 Linux 系统的引导过程；然后对 Linux 内核的 3 大核心模块——内存管理、进程管理、中断和异常处理进行了深入的分析；在此基础上，对时间度量、系统调用进行了分析和讨论；最后讲解了 Linux 内核中常见的同步机制，使读者掌握每处理器变量和 RCU 这两种新的同步机制。

本书适合 Linux 内核爱好者、Linux 驱动开发人员、Linux 系统工程师参考使用，也可以作为计算机及相关专业学生深入学习操作系统的参考书。

Linux 2.6 内核标准教程

◆ 编　　著　华清远见嵌入式培训中心　河秦　王洪涛
责任编辑　屈艳莲
执行编辑　黄　焱

◆ 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
北京鑫正大印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：23.75
字数：567 千字　　2008 年 11 月第 1 版
印数：1－4 000 册　　2008 年 11 月北京第 1 次印刷

ISBN 978-7-115-18711-6/TP

定价：49.00 元（附光盘）

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

写作背景

自由、开放的 Linux 操作系统正在蓬勃发展，得到了广泛的应用，Linux 操作系统的用户数量迅速增长，很多 Linux 爱好者希望能够学习、掌握 Linux 内核的原理、机制，能够阅读 Linux 内核代码，并能够加以应用，但在实际的学习过程中常遇到以下问题。

- Linux 内核学习门槛较高，初学者总想迅速读懂内核源代码，往往在不清楚内核运行机制和内核代码结构的情况下就开始阅读 Linux 内核代码，会遇到很大障碍。
- 随着 Linux 内核的发展，内核的代码量日益增加，系统规模不断扩大，复杂度不断提高。如何在纷繁芜杂的代码中找到自己所需的信息，是一个亟待解决的问题。
- 研究、学习过程中不仅需要掌握 C 语言、操作系统方面的知识，而且还要需要掌握汇编语言、内联汇编、编译器、链接器、链接脚本等各方面的知识。

因此，读者亟需这样一本书。

- 能够在需要的地方深入浅出地讲解研究、学习 Linux 内核所需的知识点。
- 能够对内核的核心框架进行全面剖析，引导读者走出由错综复杂的函数和数据结构组成的迷宫，使读者可以在较短的时间内掌握 Linux 内核的精髓。

本书特点

（1）基于 Linux 2.6 版本内核进行讲解。

相对于早期版本，本书讨论的 Linux 2.6 版本内核已经非常成熟，具有 O(1)调度算法、改进的 NPTL 线程模型、内核态抢占等新的特性，具有良好的响应能力（软实时）。这些重要特性保证了 Linux 2.6 版本内核具有非常广泛的实际应用价值，更适用于实际产品的开发。

（2）专注于 Linux 内核核心模块，使读者在较短的时间内掌握 Linux 的精髓。

对于 Linux 2.6 版本内核这一“庞然大物”，本书选取了内核的核心关键模块，在有限的篇幅内对 Linux 内核的工作原理进行深入、透彻的讲解。

（3）分析和讲解细致、透彻。

本书对每个内核核心模块，都深入剖析其数据结构、访问接口、工作机制和内核实现。

本书给出了 Linux 内核中每个内核模块运作机制的基本轮廓，并用示意图加以说明，帮助读者掌握相应的运作模型，并对内核实现的关键细节、关键代码进行详细分析。

本书主要内容

第 1 章首先对内核的目录结构进行了介绍，然后介绍了 Linux 2.6 内核的新特性，最后介绍了内核探索工具和阅读本书的方法。

第 2 章首先详细分析了 Linux 系统在标准 PC 上的引导过程，以及系统控制权交给内核镜像 bzImage 的过程，然后讲解内核的初始化过程，如何为第一个 C 函数设置所需的运行环境，并分析了系统初始化入口函数 start_kernel ()，使读者了解 Linux 系统最基本的初始化过程，该过程对理解其他内核模块的初始化有重要意义。

第 3 章主要讲解内存管理，这是 Linux 内核中最复杂、最核心的内核模块。本章首先介绍了 IA32 体系结构提供的内存管理机制——分段机制和分页机制，讨论了两者间的关系和 Linux 内核所做的取舍；随后详细分析了内核页表的初始化过程，并对 Linux 内核的内存模型进行了讲解，分析 Linux 是如何对 NUMA 架构提供支持的；最后讲解了物理页框的分配、回收过程，并对内核地址空间的划分和用途进行分析。

第 4 章主要讲解 Linux 内核中进程、线程的概念，对其所涉及的关键数据结构进行了讲解，分析了 Linux 内核中进程组织形式和它们各自的用途，对进程的创建过程进行了详细的讨论和分析；然后详细介绍 2.6 版本内核中新采用的 O(1)复杂度调度器的基本思想和实现细节；最后对系统中的第一个进程（0 号进程）的创建过程进行详细分析。

第 5 章所讲解的中断、异常机制是计算机系统的核心，系统调用和时间度量都是建立在该机制之上的。本章首先介绍了 IA32 体系结构的中断、异常机制，然后讲解了中断描述符表的初始化过程以及中断、异常处理过程中计算机软、硬件的工作状态和处理方法，最后讲解中断延迟问题。

第 6 章详细讲解了内核时间度量的架构和需要的硬件支持，并对时钟中断的处理过程进行分析，最后讲解 Linux 内核软定时器的工作原理和实现细节。

第 7 章首先讲解系统调用接口的作用和访问手段，然后讲解系统调用的工作机制和参数传递问题，并介绍如何向内核添加系统调用，最后讲解 IA32 体系架构引入的快速系统调用指令和 Linux 内核对其提供的支持。

第 8 章主要讲解 Linux 内核各种模块所使用的同步机制，首先讲解同步的基本原理，然后对构建在基本同步原理之上的同步机制进行了详细的讲解。

参与本书编写的人员

本书内容来源于华清远见嵌入式培训中心（http://www.farsight.com.cn/）的培训资料，是多年培训和研发经验的总结。本书主要由河秦、王洪涛负责编写，同时参与编写和资料整理的还有毛彦超、王瑞、史志宇、石玉龙、刘国明、李奘、武凡琦、陈群星、罗璋、赵炳和、崔海峰、孙琼、田旭、范文庆、钟金鑫、王欣、张曦文、尚玉珊、张丛辉、王玮、刘超、张圣亮、李凡、马堃、徐路迎、赵国锋、孙颂武、汪荷君、孙明、林雪梅、张墨、黄惠英、刘雯、张墨、郭永红、周瑜、王建伟等，在此表示衷心的感谢。

本书之外的内容

河秦是王宗涛的笔名，用以表达他和他爱人对家乡——河底、秦王寨的深厚感情，感谢家人们一年多来所给与的支持和鼓励，没有他们的支持和鼓励，很难有本书的出版。

联系我们

由于笔者水平有限、编写时间仓促，书中难免存在疏漏和不足之处，恳请广大读者提出宝贵意见。本书的相关资料和嵌入式系统相关资料、公开视频，请参见 http://www.farsight.com.cn/ download。

我们为本书开通了专门的配套网站，网址是 http://www.linuxdriver.cn/，读者可以直接同我们交流，共同学习和提高。另外，我们还为本书提供了专门的联系邮箱 qinriver@gmail.com，读者可以随时同我们联系。本书责任编辑的联系方式是 huangyan@ptpress.com.cn，欢迎来信交流。

编　者

2008 年 9 月

目　录

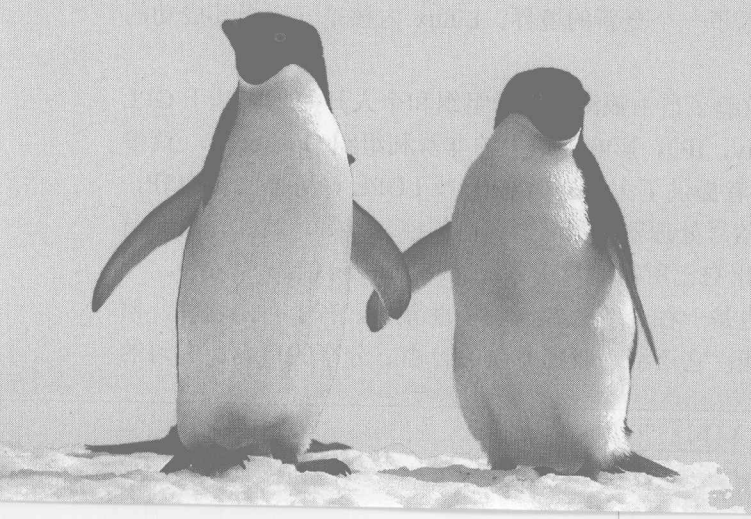
第1章 Linux内核学习基础

基于GPL（General Public License，通用公共许可证）的Linux系统在嵌入式设备中得到了广泛的应用，使得对Linux系统程序员的需求迅速增长，而且从长远来看，这一需求会不断增加。由于嵌入式系统具有高定制性，软件工程师不仅需要读懂硬件规范说明书（Hardware Specification），还要深入了解Linux内核的工作机制。这样才能编写有效的系统引导代码、系统核心初始化代码以及相关外设的驱动程序。

从什么地方入手学习Linux内核、如何学习Linux内核是每个初学者遇到的首要问题。Linux内核非常庞大和复杂，并且随着内核版本的更新，Linux内核正在变得更加复杂和庞大，因此Linux内核的学习过程是非常艰苦的。

本章主要讲解Linux内核的学习方法，如何充分利用本书提供的相关信息来加快这一学习过程，以及如何在短时间内完成对Linux内核的深层次理解。

通过本章的学习，读者将了解计算机内部如何工作、Linux内核的运行机制、Linux内核的运行前提以及所需的底层支持，掌握为新的评估板构建BSP、编写设备驱动、移植Linux内核的方法，从而成为Linux内核高手。



1.1 为什么研究 Linux 内核

本节先对 Linux 系统的来源、历史背景和现状作简单介绍，然后讨论自由软件的价值所在及自由软件如何满足用户的需要，最后讲解 Linux 系统的应用前景和未来。

1.1.1 Linux 的历史来源

谈到 Linux 内核，不能不介绍 GNU 项目。假若没有 GNU 计划和 GNU 推出的自由软件，Linux 内核不可能如此成功；Linux 内核的开发工作都是基于 GNU 推出的自由软件完成的。反过来，没有 Linux 内核，GNU 计划也只是一个空谈——它无法向用户提供一个完全自由的类 UNIX 操作系统。GNU 和 Linux 相辅相承、缺一不可。

GNU 工程始于 1984 年，由自由软件之父 Richard Stallman 组织，目的在于开发一套自由、完整的 UNIX 操作系统。该系统是一个完全“自由”的软件体系，与此相应的有一份通用公共许可证（General Public License，GPL）。和大多数软件许可证相反，GNU 通用公共许可证力图保证用户共享和修改自由软件的自由——保证自由软件对所有用户是自由的，这里指的是自由而不是免费。GPL 的详细信息，请参见 http://www.gnu.org/licenses/licenses.html。

Linux 内核及其他相关的大量软件都是在 GPL 的推动下开发和发布的。各种使用 Linux 作为内核的 GNU 操作系统正被广泛地使用着，虽然这些系统通常被称作为 Linux，但是它们应该被更确切地称为 GNU/Linux 系统。原因在于，Linux 实际上是操作系统的内核，使这个内核变得非常有用的大量应用程序都是 GNU 软件，都是在 GPL 许可证下发布的。例如，窗口管理系统、编译器、各种 shell、脚本解释器、浏览器、编辑器以及其他多不胜数的实用工具。基于该原因，GNU/Linux 更适合作为操作系统的名称，而 Linux 则适合作为内核的名称。

GNU 项目的开发策略是先完成现有 UNIX 系统中实用程序（如 ls、cat、autoconf、automake）的自由实现，然后开发 GNU 操作系统的核心——Hurd 内核，只要 Hurd 内核开发出来，GNU 就是一套真正自给自足、完整的自由操作系统。但与此同时，Linux 内核在短短几年内迅速崛起，也使用 GNU 的通用公共许可证，填补了 GNU Hurd 内核的空缺。

因为 Linux 内核已经十分成熟，目前几乎所有的 GNU 系统都采用 Linux 作为操作系统的核心。如果说 Richard Stallman 创立并推动了自由软件的发展，那么 Linus 毫不犹豫奉献给 GNU 的 Linux，则把自由软件的发展带入了一个全新的境界。Linux 内核是一个极其成功的自由软件，是自由软件中的一个典范。

除了软件行业外，硬件产业也随后掀起了自由浪潮。各种组织和个人开始发布基于 GPL 许可证的自由硬件 IP（Intellectual Property，IP），影响力最大的非营利组织 Opencores（请参阅 http://www.opencores.org）为硬件设计者提供了大量遵循 GPL 或 LGPL 许可证的自由 IP，这些 IP 包括处理器、通信控制器、数字信号处理器、存储器、视频控制器等。硬件开发人员可以利用这些自由 IP 快速地组合，构建出自己所需的片上系统（System on Chip，SoC）。

自由软件不仅仅是产品，自由软件也是一种全新的价值观和世界观。有了自由软件，信息产业就多了一个新世界、一种新的可能；它还给了我们复制的自由、分享的自由、学习的

自由；它使源代码获得解放、软件获得重生，有助于更多的人参与到软件开发中来，创造更多、更好的代码和软件，为大家所用、为人类所用。

1.1.2 Linux 的发展现状

经过近 20 年的发展，GNU/Linux 成为了一个支持多用户、多进程、多线程、实时性较好、功能强大而稳定的操作系统。它可以运行在 x86、Sun Sparc、Digital Alpha、680x0、PowerPC、MIPS、ARM 等平台上，是目前支持硬件平台最多的操作系统。除在桌面领域由于用户操作习惯等因素的制约发展不是很好外，Linux 在各个领域都取得了巨大的进步和成功。

1. 企业应用领域

在企业应用领域，Linux 得到了除微软公司之外几乎所有知名软件和硬件公司的支持。支持 Linux 的硬件公司有 IBM、HP、Sun、Intel、AMD、Sony 等，软件公司有 CA、Veritas、BEA、Oracle、SAP、Borland 等，使得 Linux 操作系统在企业运算领域站稳了脚跟。

作为一种服务器级的操作系统，Linux 非常成熟，支持多处理器、大型文件系统、日志型文件系统、密集型运算和高可用集群技术。提供 Web 服务的 Linux 系统遍布全球，而且越来越多的商业用户使用 Linux 作为文件和打印服务器。同时，Linux 作为邮件服务器的一种候选平台，也被当作安全、强壮的防火墙。

2. 嵌入式领域

随着计算机工业的发展、集成电路制造业的进步，越来越多的微处理器以各种形式、低廉的价格被制造出来，并广泛应用于各种各样的场合。这使得嵌入式系统无所不在，从早期的航空航天、国防工业、工业控制、通信设备等领域扩展到与人们生活密切相关的电子产品中。移动电话、数字电视机顶盒、游戏机、GPS 导航设备等家用电器彻底改变了人们的生活方式。

其他嵌入式操作系统如 VxWorks、pSOS、Neculeus 和 Windows CE 等都是商业化产品，其高昂的价格使许多生产低端产品的小公司望而却步；而且，源代码封闭性也大大限制了其产品对新型处理器、开发平台的支持力度。因此，自由、开放的 Linux 操作系统在此机会下蓬勃发展。由于 Linux 在嵌入式操作系统市场的良好表现，使得世界上最大的面向嵌入式设备软件供应商之一的美国风河系统公司开始全面支持嵌入式 Linux 系统，提高相关产品和服务。

1.1.3 Linux 的前景展望

Linux 和嵌入式 Linux 在过去几年里已经越来越普遍地被 IT 业、半导体公司、嵌入式系统所认可和接受，它已经成为一个可以替代 Windows 和众多传统 RTOS（实时操作系统）的重要的操作系统。Linux 因其强壮、稳定、网络功能强大的特点而在服务器领域极为流行；在嵌入式设备领域，如信息电器、机顶盒、防火墙和路由器市场上，Linux 的应用前景广阔。

1.2 选择什么版本进行研究

广泛应用于当今社会经济活动中的 Linux 操作系统，给人类创造了巨大的物质财富，是

人类知识和智慧的结晶，因此学习和研究 Linux 内核具有重要的实际意义。学习、研究 Linux 内核不仅可以巩固和加深对操作系统的理论理解，还可以将这些知识应用到实际工作中去，指导工程项目的开发过程，为社会创造价值。

本书从实用的角度出发，选择 Linux 2.6.15.5 内核作为研究对象。虽然低版本的内核更容易讨论、更容易理解，能够较快地掌握操作系统运行机制，但是这些低版本的内核缺乏一些重要特性和实际的应用价值。本书讲解的 Linux 2.6.15.5 内核是一个成功的、企业级的操作系统核心，具有 O(1)调度算法、改进的 NPTL 线程模型、内核态抢占等新特性，以及良好的响应能力（软实时）。

因为选择新版本内核进行讲解，所以不能对所有的内核模块逐一进行详尽的分析，必须做出取舍。本书选择了内核最核心的几个模块进行详细讲解，对每个模块的核心数据结构、访问接口、工作机制、内核实现进行了深入讨论和剖析。

1.3 内核基本结构

操作系统是对计算机系统的硬件和软件资源进行组织和管理的系统软件，用于为用户提供相关的服务和访问接口，它是计算机系统中最底层、最基础的支撑软件。其中，操作系统最核心的部分为内核，其功能包括：处理机调度、内存管理、进程管理、设备管理、文件管理等。其目的在于把硬件裸机改造成为功能完善的一台虚拟机，使得计算机系统的使用和管理更加方便，使计算机系统中所有资源的利用率更高，并为用户提供方便、高效、友善的操作界面。

1.3.1 内核在操作系统中的地位

GNU/Linux 操作系统由一套基于 GPL 许可证的软件组成，其中 Linux 是该操作系统的核心，其他软件则作为该操作系统的核心组件（如 C 运行库 glibc、窗口管理器 Gnome、命令解释器 shell 等）和各种各样的应用程序。系统硬件、系统核心和其他应用软件三者的关系如图 1.1 所示。

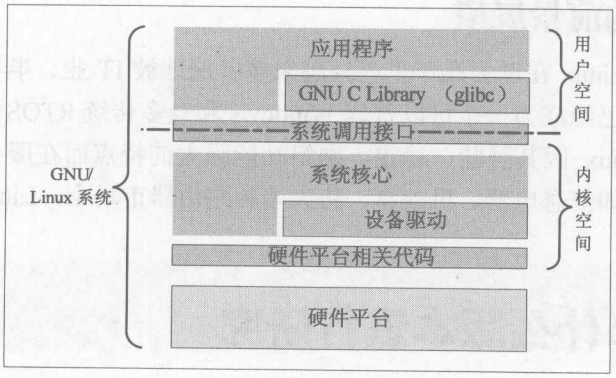


图 1.1 内核在操作系统中的地位

从图 1.1 中可以看出，应用程序直接或间接（通过 glibc 的封装函数）调用系统调用接口来获得内核提供的服务。系统调用接口在这里作为一个“门”存在，通常称之为系统调用门，它是系统由用户态进入内核态的一种途径。内核态时可以执行特权指令，用户态则没有此权限。所有服务请求在通过该门的时候，都要做安全检查以保证系统的安全。

除了提供系统调用门为用户进程服务之外，内核还提供了中断门用于响应外设的任务处理请求，中断门是系统进入内核态的另外一种途径。异常门是系统进入内核态的另一种途径。根据执行路径的特性（进程对应于一个进程描述符，执行路径不需要有进程描述符，也就是说，进程一定是执行路径，但是执行路径不一定是进程），我们可以将系统中的执行路径分为以下几种。

- 用户进程运行于用户态，此时系统处于用户态，这也是最常见的状态，任何一个应用程序（称为“进程”更确切一点，因为一个应用程序可以拥有多个进程，也就是多个执行路径）通常都处于此种状态之下。
- 用户进程运行于内核态，当系统处于内核态，一个应用程序发出系统调用请求之后，系统即进入内核态，执行内核态的系统调用处理函数，为用户进程服务。
- 内核线程运行于内核态，此时系统处于内核态，内核线程是一种特殊的进程，它没有用户地址空间（线性地址在 0xC0000000～0xFFFFFFFF 之间称为内核地址空间，线性地址在 0x00000000～0xBFFFFFF 之间称为用户地址空间），永久运行于内核态。
- 系统处于中断上下文中，此时系统处于内核态，为外设的中断处理请求服务，与系统中的进程没有任何联系，通常称此时的状态为系统处于中断上下文之中（包括中断上半部和除了工作队列之外的所有中断下半部）。

1.3.2 Linux 2.6 内核源代码目录树简介

在分析内核之前，如果能从全局对 Linux 内核的目录结构有一个整体印象，不仅有利于内核分析过程，而且通过了解内核源码的布局，我们可以了解大型项目代码布局的规则和经验，使整个项目更加有条理，从而了解 Linux 移植、添加驱动时需修改哪些目录的文件，哪些不需修改，以提高读者的实际动手能力。下面对 Linux 2.6.15.5 内核的源代码目录结构进行简单的介绍，代码树如图 1.2 所示。

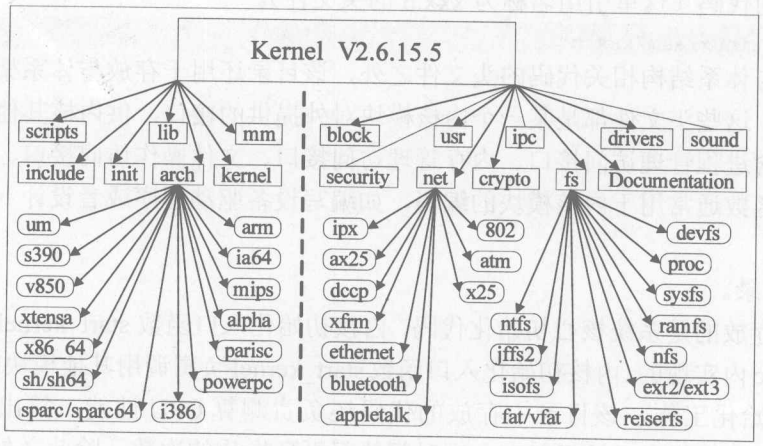


图 1.2 内核代码目录树

1. 系统最核心组件

图 1.2 中的左侧部分是本书讲解的重点，这些目录对应的模块是 Linux 内核运行不可缺少的重要组成部分，是操作系统最核心的部分。通过图中 arch 子目录我们可以看出 Linux 内核支持多种体系架构，从常见的 i386 到销声匿迹的 Alpha 架构；从 32 位处理器到 64 位处理器；从广泛应用于嵌入式系统的 ARM 架构到专门应用于高性能计算的 Power 架构（PowerPC 中的一种）。另外还提供了一个软的架构——um，通过该架构，用户可以在 Linux 系统中运行另外一个 Linux 系统。本书只讨论 Linux 在 i386 架构上的实现。

（1）scripts 目录。

该目录中不包含任何核心代码，该目录下存放了用来配置内核的脚本和应用程序源码。当输入 make menuconfig 或 make xconfig 之类的命令进行内核配置时，系统会首先编译应用程序的源码，生成可执行文件，然后运行。该可执行文件接着读取当前体系结构对应子目录树中的 Kconfig 文件开始对内核进行配置，在配置完成后会在当前内核源代码根目录中生成一个.config 文件。除此之外，这些脚本还负责提取可执行内核镜像 vmLinux 的符号表信息，存入文件 System.map 中，供内核调试使用。

（2）lib 目录。

该目录主要包含两部分内容：gnuzip 解压缩算法，用于在系统启动过程中将压缩的内核镜像解压缩；剩余的文件用于实现一个 C 库的子集，主要包括字符串和内存操作等相关函数。

（3）mm 目录。

该目录包含了体系结构无关的内存管理代码，包括通用的分页模型的框架、伙伴算法的实现和对象缓冲器 slab 的实现代码。这些代码最终依赖于具体的平台完成相关操作。

（4）include 目录。

这个目录包含了 Linux 源代码目录树中绝大部分头文件，每个体系架构都在该目录下对应一个子目录，该子目录中包含了给定体系结构所必需的宏定义和内联函数。这些内联函数很多都是全部或部分地使用汇编语言实现的，而且在 C 或汇编代码中都会应用到这些文件。在编译内核的过程中，首先会建立一个名称为 include/asm 的符号链接，该链接指向目标体系结构在该目录中的子目录，使得与体系结构无关内核代码可以使用如下简洁的形式来引用与体系结构相关的代码（这里引用名称为 xxx.h 的头文件）。

```
#include "Linux/asm/xxx.h"
```

除了包含与体系结构相关代码的头文件之外，该目录还用于存放与体系架构无关的内核模块的头文件，这些头文件都是某一个内核模块对外提供的接口，供内核其他模块使用。这些接口主要包含进程管理访问接口、内存管理访问接口、文件操作访问接口、网络访问接口等，这些接口函数通常用于内核模块的编写，如编写设备驱动程序或者设计一个新的文件系统等。

（5）init 目录。

该目录中存放的是系统核心初始化代码，内核初始化入口函数 start_kernel 就是在该目录中的文件 main.c 内实现的。内核初始化入口函数 start_kernel 负责调用其他模块的初始化函数，完成系统的初始化工作。该目录中存放的代码还负责测算 CPU 性能，算出系统处理器的 BogoMIPS 值，该值代表了处理器在一定时间执行循环操作的次数。除此之外，该目录中的

代码还负责完成根文件系统初始化过程中的相关工作。

（6）kernel 目录。

该目录中存放的是 Linux 内核的最核心代码，用于实现系统的核心模块，这些模块包括：进程管理、进程调度器、中断处理、系统时钟管理、同步机制等。该目录中的代码实现了这些核心模块的主体框架，独立于具体的平台和体系架构。核心模块与平台相关代码存放在目录 arch 中，具体请看后面对该目录的介绍。

（7）arch 目录。

该目录中的每个子目录中都与某种体系结构相对应，用于存放体系架构相关代码，向平台无关的系统核心模块提供所需的功能接口。每个体系结构对应的子目录下通常至少包含以下几个子目录。

- kernel 子目录：用于存放特定体系结构特有信号量的实现代码和对称多处理器（Symmetric MultiProcessing，简称 SMP）相关模块。
- lib 子目录：用于存放依赖于当前特定体系结构的辅助功能，如利用当前体系结构特性实现的 strlen 和 memcpy 内存操作函数；与通用的实现方法相比，它们的开销小、更加高效。
- mm 子目录：用于存放体系结构特定的内存管理模块，包括内存的初始化、页表管理等内容。
- boot 子目录：该目录中包含了当前平台上系统引导过程使用的部分或全部代码。这部分代码依赖于当前平台，用于完成向系统内存装载内核镜像的工作。

2．系统次核心组件

图 1.2 中右侧部分是内核的其他模块，这些模块用于实现相对不是很重要的内核模块。从 net 和 fs 目录可以看出 Linux 内核支持众多的文件系统和网络协议，这也是 Linux 取得成功的原因之一。下面对这些目录的用途、功能进行简要的介绍。

（1）block 目录。

该目录用于实现块设备的基本框架和块设备的 I/O 调度算法。

（2）usr 目录。

该目录中的代码为内核尚未完全启动时执行用户空间代码提供了支持。

（3）ipc 目录。

该目录中的文件用于实现 System V 的进程间通信（Inter Process Communication，IPC）模块。

（4）driver 目录。

该目录用于存放各类设备的驱动程序。

（5）sound 目录。

该目录存放了声音系统架构，如 Open Sound System（OSS）、Advanced Linux Sound Architecture（ALSA）的相关代码和具体声卡的设备驱动程序。

（6）secturity 目录。

该目录中存放了 Security-Enhanced Linux（简称 SELinux）安全框架的实现代码。

（7）crypto 目录。

该目录中存放了相关的加密算法的代码。

（8）Documentation 目录。

该目录中存放了与内核相关的为数不多的文档，包括一些相当优秀而且相当完整的文档。

1.3.3 Linux 2.6 内核的新特性

无论是对于企业服务器还是对于嵌入式系统，Linux 2.6 内核都是一个巨大的进步。对高端服务器而言，主要改进包括对 SMP 系统和非一致性内存（Non-Uniform Memory Access，NUMA）系统的支持。另外采用了新的线程模型 NPTL、每处理器变量、O(1)复杂度的调度算法，提高了系统的可扩展性和性能。对嵌入式领域而言，增加了对新的体系结构和处理器类型（包括没有内存控制器）的支持；为了提高桌面用户的满意度和系统的实时能力，增加了对内核态抢占的支持，同时借助于更高的时钟粒度，提高了系统的响应能力和实时性。下面对 Linux 2.6 内核的新特性进行简要的介绍。

1．新的调度算法

Linux 2.6 内核采用了新的调度算法，其时间复杂度为 O(1)，它在高负载的情况下表现得极其出色。该调度器的主要目标是高效、公平地分配 CPU 时间，同时提供很好的用户体验。为此，该调度算法需要处理面对一些互相冲突的需求，例如既要为关键实时任务最小化响应时间，又要最大限度地提高系统的吞吐量。总地来说 Linux 2.6 内核的调度器实现了以下目标（新调度器的详细介绍，请参见本书 4.5 节）。

（1）时间复杂度为 O(1)，即不管系统中当前有多少进程，选择下一个合适进程投入运行的所需时间不变，即新调度算法的复杂度与问题的规模无关，具有良好的伸缩性和扩展性。

（2）实现对多处理器系统的良好支持，系统中的每个处理器都拥有自己的运行队列，具有良好的可扩展性；同时强化了 SMP 系统中的 CPU 亲和力，尽量将一组相关的进程分配给同一个 CPU 进行处理。有利于优化处理器缓存的性能，同时解决了多处理器上的负载平衡问题。

（3）提供了良好的交互能力，即使在系统负载相当重的情况下，也能保证对用户输入的快速响应。

2．新的线程模型

最初的 Linux 设计并不能支持线程编程模型。但是它可以通过系统调用 clone()创建父进程的一个拷贝，这个拷贝与调用进程共享相同的地址空间，这样，满足了线程之间共享同一地址空间的要求。LinuxThreads 线程库基于这个系统调用来实现线程模型，该线程模型借助一个名为“管理线程”的管理者在用户空间模拟对线程的支持。这种方法有一些缺点，尤其是在信号处理和进程间同步方面存在严重的问题。另外，这个线程模型也不符合 POSIX 的要求。该线程模型的缺陷如下。

（1）它使用管理线程来创建线程，并对每个进程拥有的所有线程进行协调，这增加了创建和销毁线程所需的开销。

（2）由于管理线程只能在一个 CPU 上运行，因此执行的同步操作在 SMP 或 NUMA 系统上可能会产生可伸缩性的问题，即该线程模型的可扩展性差。

（3）由于线程的管理方式，以及每个线程都使用了一个不同的进程 ID，因此 LinuxThreads