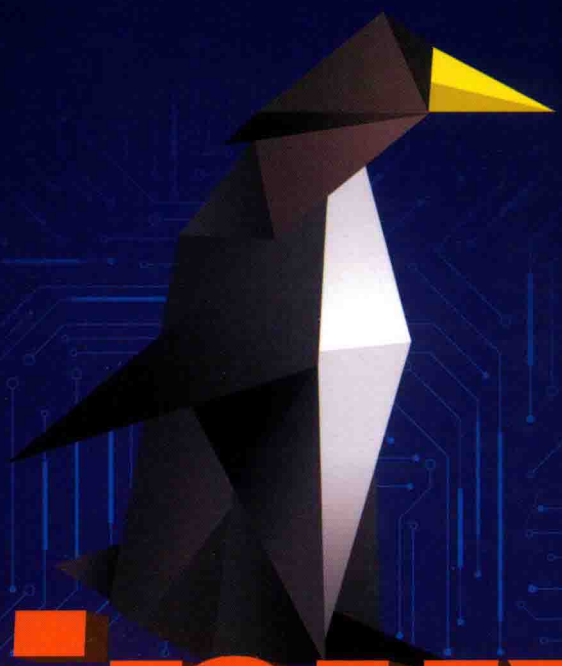
异步图书
www.epubit.com

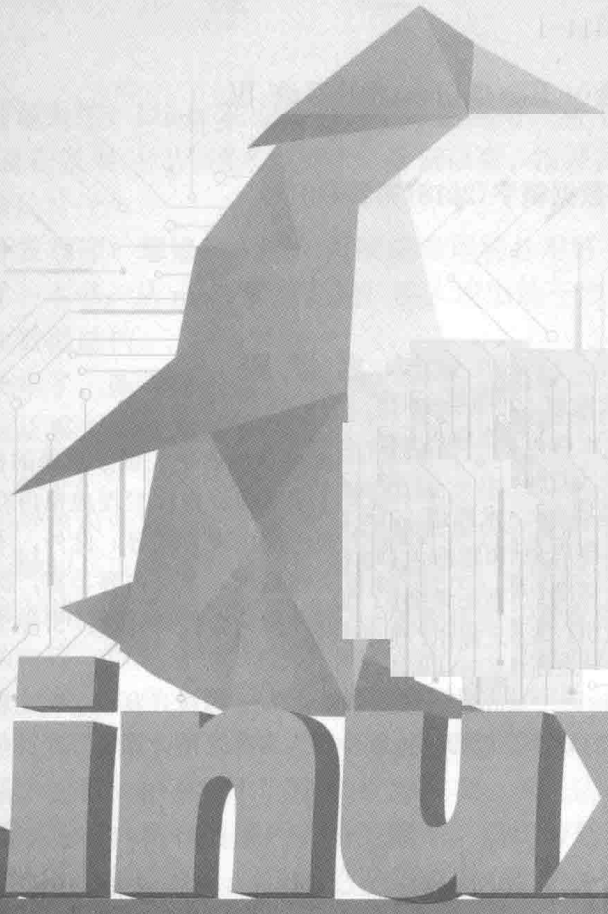


Linux 内核深度解析

基于ARM64架构的Linux 4.x内核

余华兵 著

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS



Linux 内核深度解析

余华兵 著

人民邮电出版社
北京

图书在版编目（CIP）数据

Linux内核深度解析 / 余华兵著. -- 北京 : 人民邮电出版社, 2019.5
ISBN 978-7-115-50411-1

Ⅰ. ①L… Ⅱ. ①余… Ⅲ. ①Linux操作系统 Ⅳ. ①TP316.85

中国版本图书馆CIP数据核字(2018)第294431号

内 容 提 要

本书基于4.x版本的Linux内核，介绍了Linux内核的若干关键子系统的技术原理。本书主要内容包括内核的引导过程、内核管理和调度进程的技术原理、内核管理虚拟内存和物理内存的技术原理、内核处理异常和中断的技术原理，以及系统调用的实现方式等。此外，本书还详细讲解了内核实现的各种保护临界区的互斥技术，以及内核的文件系统。本书内容丰富，深入浅出，通过大量的图例来描述数据结构之间的关系和函数的执行流程，并结合代码分析，引导读者阅读和理解内核源代码。

本书适用于负责维护和开发Linux内核或基于Linux内核开发设备驱动程序的专业人士，以及想要学习了解Linux内核的软件工程师，也适合作为高等院校计算机专业的师生用书和培训学校的教材。

◆ 著　　余华兵
责任编辑　张　爽
责任印制　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京天宇星印刷厂印刷

◆ 开本：787×1092　1/16
印张：39.75
字数：972千字　　2019年5月第1版
印数：1－2 400册　　2019年5月北京第1次印刷

定价：138.00元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字20170147号

序　一

作为一个优秀的开源系统，Linux 在业界有很广泛的应用。从 1994 年发布 1.0 正式版本开始，Linux 内核一直在发展，代码越来越庞大，参伍以变，错综其数，要想深入掌握它，并不是一件轻松的事情。

对于在 Linux 上开发程序、想修改 Linux 内核或者理解并用好 Linux 的软件开发工程师而言，迫切需要这样一本书：从 main 函数开始，深入浅出地一步步剖析 Linux 内核，并解释其中的函数关系和数据结构。

我与华兵共事十余年了，两年前听说他开始写一本关于 Linux 内核的书，我就一直期待这本书的出版。华兵从事 Linux 内核开发工作十余年，有着丰富的实践经验，使用过不同的版本和硬件平台，从 2.x 到 4.x，从 MIPS、x86 到 ARM。伴随着研发大楼的华灯，当年初出校门的学子，如今萧萧两鬓生华，成为一个在 Linux 领域有深厚积累的专家。更难得的是，他愿意将这些经验写成一本书，与同道中人共享，并醉心于此，力求至简。

当然，我们在市场上陆续能看到关于 Linux 内核的技术著作，各有特点，其中也不乏经典。不少作者和华兵一样，既是 Linux 技术的爱好者，也是长期的实践者，并将知识与经验提炼成文，与读者分享。与已上市的 Linux 图书相比，《Linux 内核深度解析》在内容上有其独特之处。

本书剖析的代码基于 Linux 4.12 版本，发布于 2017 年，是 Linux 内核史上变动较大的版本之一。基于这个版本进行内核代码解析并出版成书，是比较新颖的，既不失通用性，又兼顾 4.x 版本中引入的不少新技术点。同时，它基于 ARM64 硬件平台，将两者结合的书，目前还是比较少的。

另外，本书没有过多地介绍操作系统的基础原理，而更多地是以实际代码来解读在 Linux 内核中是如何实现操作系统的各个子系统的。对于熟悉操作系统基础的读者来说，可以快速地切入到具体代码的理解与实现中。从内核引导和初始化开始，到进程管理、内存管理、中断/异常/系统调用、内核互斥技术和文件系统，本书比较系统地对内核代码进行了深度解析。

Linux 内核的知识点相当繁多，很难在一本书中面面俱到，也没有必要。所以，在这本书中，看似不经意间逐层展开的知识点，都是比较基础和常用的。作者以他的实践经历尽量通俗地进行解读，并抓住了其中的重点，可以让读者在实际的开发、调试和维护工作中学以致用。

“行是知之始，知是行之成”，学习 Linux 内核技术尤其如此。要真正消化理解 Linux 内核，离不开大量的工程实践。希望本书可以成为你前进路上的好帮手！

锐捷网络研究院副院长　林东豪

2019 年 2 月

序　二

我很荣幸接受华兵的邀请，为他的新书作序。

作为国内数通厂商的一位资深人士，华兵经历了数通厂商操作系统的大变化。早些年数通领域各厂商（包括思科和华为）的操作系统，都是基于传统的嵌入式操作系统（如典型的 VxWorks 操作系统）开发的。2010 年以后，Linux 内核在数通厂商中快速生长，迅速成为数通设备网络操作系统的内核。就像基于 Linux 内核的安卓系统已经成为智能手机领域的领头羊一样，在数通设备领域，Linux 内核也大有一统江湖之势。

在此背景之下，华兵作为公司最早一批参与基于 Linux 内核的网络操作系统开发的工程师，积累了丰富的经验，还向 Linux 开源社区提交其发现的多个问题。

我们在基于 Linux 内核开发网络操作系统的过程中遇到的某些技术问题，在 Linux 内核的演进过程中已经提供了解决方案。Linux 3.11 版本 ARM 架构支持巨型页机制，解决了 ARM 架构的进程访问大内存的性能问题。Linux 3.14 版本引入 ZRAM 内存压缩技术，用于节省内存空间，这项技术适合在内存容量小的设备上使用。Linux 2.6.29 版本引入的 squash 文件系统和 Linux 3.18 版本引入的 overlay 文件系统，在闪存容量小的设备上解决了存储空间不足的问题。squash 文件系统可以压缩数据，但是它是一个只读的文件系统，而设备需要一个可写的文件系统，我们在 Linux 内核找到了解决方案——使用 overlay 文件系统在 squash 文件系统上面叠加一个可写的文件系统。这些拿来即用的 Linux 内核技术，在华兵的这本书中都有提及。

近年来 Linux 发展迅速，公司最早使用的 Linux 内核是 2.6 版本，从 2011 年发布 3.0 版本开始到 2018 年年底发布 4.20 版本，Linux 一共发布了 41 个版本，技术发展日新月异。很多技术虽然有众多的工程师在使用和总结，但仍相对零散，基于 Linux 4.x 的图书更是少之又少。

作为内核技术的深度爱好者，华兵萌生了写一本 Linux 内核技术著作的想法，以此记录自己的学习心得，分享自己的工程解决方案，也帮助更多的人学习和使用 Linux。鉴于工作原因，华兵在 MIPS 架构上有着很丰富的工作经验，但他在写作中选择了有着更加广泛使用基础的 ARM64 架构，并对基于 ARM64 架构的代码进行分析。

写书的过程是很辛苦的，每个夜晚和周末，他都放弃休息，用来分析源代码、查阅资料、埋头写作，坚持一年多才得以成书。向不善言谈的华兵致敬，向每个热爱技术的人致敬，他们都是可爱的人！相信本书能对从事 Linux 研究和使用内核技术的人提供很好的帮助。

锐捷网络研究院软件平台总监　黄崇滨

2019 年 3 月

前　言

Linux 内核是使用最广泛的开源内核，在服务器和智能手机领域处于统治地位，物联网、大数据、云计算和人工智能等热点技术也离不开 Linux 内核。对于商业公司而言，采用开源的 Linux 内核可以享受很多好处，比如节约成本，可以利用行业先进的技术，还可以根据自己的需求定制、修改内核。对于个人而言，从 Linux 内核中可以学习先进的设计方法和编程技术，为内核贡献代码可以证明自己的技术实力。

可是，当我们准备学习 Linux 内核时，会发现 Linux 内核的代码庞大而复杂，在没有专业书籍指导的情况下，读懂代码是一件非常困难的事情。作者编写本书的目的是为想要深入理解 Linux 内核的软件工程师提供指导。

本书介绍 4.12 版本的 Linux 内核，建议读者在阅读本书时到 Linux 内核的官方网站中下载一份代码，对照代码学习。推荐使用“Source Insight”软件阅读代码。

Linux 内核支持多种处理器架构，处理器架构特定的代码放在“arch”目录下。ARM 处理器在手机和平板电脑等移动设备上处于统治地位。ARM 处理器从 ARMv7 演进到支持 64 位的 ARMv8，ARM 公司重新设计了处理器架构，ARMv8 定义了 AArch64 和 AArch32 两种执行状态，AArch64 是 64 位架构；AArch32 是 32 位架构，兼容 ARMv7。因为 ARMv8 和 ARMv7 的差别很大，所以 Linux 内核把 ARMv8 和 ARMv7 当作两种不同的处理器架构，ARMv7 架构的代码放在“arch/arm”目录下，ARMv8 架构的代码放在“arch/arm64”目录下。人们通常把 ARMv8 架构的 AArch64 执行状态称为 ARM64 架构。本书在介绍 Linux 内核时选择 ARM64 处理器架构。

学习本书，需要具备 ARM64 处理器的基础知识，推荐以下两篇文档，读者可以从 ARM 公司的网站下载。

（1）“ARM Cortex-A Series Programmer’s Guide for ARMv8-A”：这篇文档接近 300 页，适合入门学习。

（2）“ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile”：这篇文档有 6000 多页，写得很详细，适合当作工具书来查询。

学习内核，关键是要理解数据结构之间的关系和函数调用关系。内核中数据结构之间的关系错综复杂，函数调用层次深，有些函数中的分支非常多，一个函数就可能涉及很多技术，这些都是初学者学习中的障碍。作者建议读者在学习时抓住主要线索，弄清楚执行流程，刚开始不要过多关注函数的细节。为了方便学习，作者绘制了很多图来描述数据结构之间的关系和函数的执行流程。另外，作者在介绍每种技术时会先介绍使用方法，从使用方法开始学习技术，相信会对读者理解技术有很大的帮助。

全书内容共分为 6 章。

第 1 章介绍内核的引导过程，本书选择常用的引导程序 U-Boot，读者可以从德国 DENX 软件工程中心的网站下载 U-Boot 的代码，对照学习。

第 2 章介绍内核管理和调度进程的技术原理。

第 3 章介绍内核管理虚拟内存和物理内存的技术原理。

第 4 章介绍内核处理异常和中断的技术原理，以及系统调用的实现方式。

第 5 章介绍内核实现的各种保护临界区的互斥技术。

第 6 章介绍内核的虚拟文件系统，内核使用虚拟文件系统支持各种不同的文件系统。

本书适用于维护或者开发 Linux 内核的软件工程师、基于 Linux 内核开发设备驱动程序的软件工程师，以及想要学习了解 Linux 内核的软件工程师和学生。

对于从事应用程序开发的软件工程师，是否有必要学习内核呢？应用程序通常使用封装好的库，看起来似乎和内核没有关系，但是库是在内核提供的系统调用的基础上做了一层封装。读者如果研究了库函数和内核配合实现库函数提供的功能，那么对软件运行过程的理解将会更深刻，个人的技术水平也将会提升到新的高度——能够设计开发出高质量的应用程序，在软件运行过程中出现问题时可以快速地分析定位。另外，内核代表了软件行业的最高编程技术，这些编程技术也适用于应用程序。

最后，感谢我的家人和朋友在本书编写过程中提供的大力支持，感谢提供宝贵意见的同事们，感谢提供技术支持的朋友们，感恩我遇到的众多良师益友。

余华兵

2019 年春

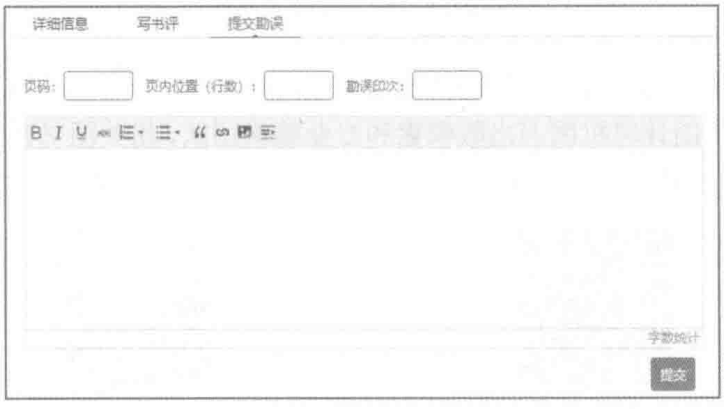
资源与支持

本书由异步社区出品，社区（https://www.epubit.com/）为您提供相关资源和后续服务。

提交勘误

作者和编辑尽最大努力来确保书中内容的准确性，但难免会存在疏漏。欢迎您将发现的问题反馈给我们，帮助我们提升图书的质量。

当您发现错误时，请登录异步社区，按书名搜索，进入本书页面，点击“提交勘误”，输入勘误信息，点击“提交”按钮即可。本书的作者和编辑会对您提交的勘误进行审核，确认并接受后，您将获赠异步社区的 100 积分。积分可用于在异步社区兑换优惠券、样书或奖品。



扫码关注本书

扫描下方二维码，您将会在异步社区微信服务号中看到本书信息及相关的服务提示。

与我们联系

我们的联系邮箱是 contact@epubit.com.cn。

如果您对本书有任何疑问或建议，请您发邮件给我们，并请在邮件标题中注明本书书名，以便我们更高效地做出反馈。

如果您有兴趣出版图书、录制教学视频，或者参与图书翻译、技术审校等工作，可以发邮件给我们；有意出版图书的作者也可以到异步社区在线提交投稿（直接访问 www.epubit.com/selfpublish/submission 即可）。

如果您是学校、培训机构或企业，想批量购买本书或异步社区出版的其他图书，也可以发邮件给我们。

如果您在网上发现有针对异步社区出品图书的各种形式的盗版行为，包括对图书全部或部分内容的非授权传播，请您将怀疑有侵权行为的链接发邮件给我们。您的这一举动是对作者权益的保护，也是我们持续为您提供有价值的内容的动力之源。

关于异步社区和异步图书

“异步社区”是人民邮电出版社旗下 IT 专业图书社区，致力于出版精品 IT 技术图书和相关学习产品，为作译者提供优质出版服务。异步社区创办于 2015 年 8 月，提供大量精品 IT 技术图书和电子书，以及高品质技术文章和视频课程。更多详情请访问异步社区官网 https://www.epubit.com。

“异步图书”是由异步社区编辑团队策划出版的精品 IT 专业图书的品牌，依托于人民邮电出版社近 30 年的计算机图书出版积累和专业编辑团队，相关图书在封面上印有异步图书的 LOGO。异步图书的出版领域包括软件开发、大数据、AI、测试、前端、网络技术等。

异步社区

微信服务号

目　录

第1章 内核引导和初始化

处理器上电以后，首先执行引导程序，引导程序把内核加载到内存，然后执行内核，内核初始化完成以后，启动用户空间的第一个进程。

1.1 到哪里读取引导程序

处理器到哪里读取引导程序的指令？处理器在上电时自动把程序计数器设置为处理器厂商设计的某个固定值，对于 ARM64 处理器，这个固定值是 0。处理器的内存管理单元（Memory Management Unit，MMU）负责把虚拟地址转换为物理地址，ARM64 处理器刚上电的时候没有开启内存管理单元，物理地址和虚拟地址相同，所以 ARM64 处理器到物理地址 0 取第一条指令。

嵌入式设备通常使用 NOR 闪存作为只读存储器来存放引导程序。NOR 闪存的容量比较小，最小读写单位是字节，程序可以直接在芯片内执行。从物理地址 0 开始的一段物理地址空间被分配给 NOR 闪存。

综上所述，ARM64 处理器到虚拟地址 0 取指令，就是到物理地址 0 取指令，也就是到 NOR 闪存的起始位置取指令。

1.2 引导程序

嵌入式设备通常使用 U-Boot 作为引导程序。U-Boot（Universal Boot Loader）是德国 DENX 软件工程中心开发的引导程序，是遵循 GPL 条款的开源项目。

下面简要介绍 ARM64 处理器的 U-Boot 程序的执行过程，入口是文件“arch/arm/cpu/armv8/start.S”定义的标号_start，我们从标号_start 开始分析。

1.2.1 入口_start

标号_start 是 U-Boot 程序的入口，直接跳转到标号 reset 执行。

arch/arm/cpu/armv8/start.S

```
1   .globl     _start
2   _start:
3    b     reset
```

1.2.2　标号 reset

从标号 reset 开始的代码如下：

arch/arm/cpu/armv8/start.S

```
1   reset:
2    /* 允许板卡保存重要的寄存器*/
3    b   save_boot_params
4   .globl    save_boot_params_ret
5   save_boot_params_ret:
6
7   #ifdef CONFIG_SYS_RESET_SCTRL
8    bl reset_sctrl
9   #endif
10   /*
11    * 异常级别可能是3、2或者1，初始状态:
12    * 小端字节序，禁止MMU，禁止指令/数据缓存
13    */
14    adr  x0, vectors
15    switch_el x1, 3f, 2f, 1f
16   3:    msr  vbar_el3, x0
17    mrs  x0, scr_el3
18    orr  x0, x0, #0xf              /* 设置寄存器SCR_EL3的NS、IRQ、FIQ和EA四个位 */
19    msr  scr_el3, x0
20    msr  cptr_el3, xzr             /* 启用浮点和SIMD功能*/
21   #ifdef COUNTER_FREQUENCY
22    ldr  x0, =COUNTER_FREQUENCY
23    msr  cntfrq_el0, x0            /* 初始化寄存器CNTFRQ */
24   #endif
25    b    0f
26   2:    msr   vbar_el2, x0
27    mov  x0, #0x33ff
28    msr  cptr_el2, x0              /* 启用浮点和SIMD功能 */
29    b    0f
30   1:    msr    vbar_el1, x0
31    mov  x0, #3 << 20
32    msr  cpacr_el1, x0             /* 启用浮点和SIMD功能 */
33   0:
34    …
35
36    /* 应用ARM处理器特定的勘误表*/
37    bl   apply_core_errata
38
39    /* 处理器特定的初始化*/
40    bl   lowlevel_init
41
42   #if defined(CONFIG_ARMV8_SPIN_TABLE) && !defined(CONFIG_SPL_BUILD)
43    branch_if_master x0, x1, master_cpu
44    b    spin_table_secondary_jump
45    /* 绝对不会返回*/
46   #elif defined(CONFIG_ARMV8_MULTIENTRY)
47    branch_if_master x0, x1, master_cpu
48
49    /*
50     * 从处理器
51     */
52   slave_cpu:
53    wfe
54    ldr  x1, =CPU_RELEASE_ADDR
55    ldr  x0, [x1]
```