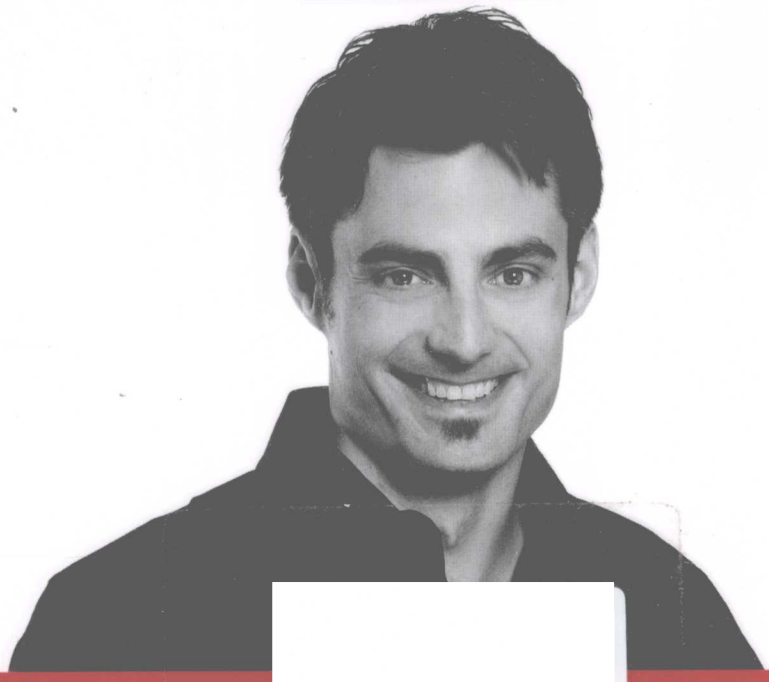
TURING 图灵程序设计丛书 Linux/UNIX系列

WILEY



Professional Linux Kernel Architecture

深入Linux内核架构

[德] Wolfgang Mauerer 著
郭旭 译

- 内容全面深入
- 全球开源社区集体智慧结晶
- 领略Linux内核的绝美风光

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 Linux/UNIX系列

Professional Linux Kernel Architecture

深入Linux内核架构

[德] Wolfgang Mauerer 著
郭旭 译

人民邮电出版社
北 京

图书在版编目（CIP）数据

深入Linux内核架构 /（德）莫尔勒（Mauerer,W.）著；郭旭译. -- 北京：人民邮电出版社，2010.6
（图灵程序设计丛书）
书名原文：Professional Linux Kernel Architecture
ISBN 978-7-115-22743-0

Ⅰ. ①深… Ⅱ. ①莫… ②郭… Ⅲ. ①Linux操作系统
Ⅳ. ①TP316.89

中国版本图书馆CIP数据核字(2010)第065664号

内 容 提 要

本书讨论了Linux内核的概念、结构和实现。主要内容包括多任务、调度和进程管理，物理内存的管理以及内核与相关硬件的交互，用户空间的进程如何访问虚拟内存，如何编写设备驱动程序，模块机制以及虚拟文件系统，Ext文件系统属性和访问控制表的实现方式，内核中网络的实现，系统调用的实现方式，内核对时间相关功能的处理，页面回收和页交换的相关机制以及审计的实现等。此外，本书借助内核源代码中最关键的部分进行讲解，帮助读者掌握重要的知识点，从而在运用中充分展现Linux系统的魅力。

本书适合Linux内核爱好者阅读。

图灵程序设计丛书
深入Linux内核架构

◆ 著 [德] Wolfgang Mauerer
译 郭 旭
责任编辑 傅志红
执行编辑 印星星 杨 爽

◆ 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
北京艺辉印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：66
字数：1852千字 2010年6月第1版
印数：1-3 500册 2010年6月北京第1次印刷
著作权合同登记号 图字：01-2009-5737号
ISBN 978-7-115-22743-0

定价：149.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

致　谢

首先，我要感谢多年以来建立了Linux内核的数千程序员，虽然他们大多数都受雇于商业公司，但也有一些人只是出于个人爱好或学术兴趣。没有这些人，就没有所谓的内核，我也没有什么可写的。请恕我无法在此列出这数千人的姓名，但按照真正的UNIX行事风格，读者很容易通过下述代码生成所有人姓名的列表：

```
for file in $ALL_FILES_COVERED_IN_THIS_BOOK; do
        git log --pretty="format:%an" $file; done |
sort -u -k 2,2
```

我非常佩服这些程序员所做的工作，他们是本书真正的英雄！

本书的演变发展已经超过7年。第一版的写作经历了两年，最终在2003年由Carl Hanser Verlag出版（德文版）。第一版讲述了2.6.0版本的内核。由于在对Red Hat Enterprise Linux 5进行EAL4+安全评估时，使用本书第一版作为底层设计文档的蓝本，因此要求更新书的内容到内核版本2.6.18。（如果读者不太明白EAL缩写的意思，可以参考维基百科）。惠普公司赞助将本书第一版翻译为英文，并授权出版英文版。接下来，我又特地将本书的内容更新到内核版本2.6.24。

有若干人士参与了本书的演变过程，向他们表示感谢。Leslie Mackay-Poulton在David Jacobs的帮助下，将本书德文版翻译为英文，完成了一件艰巨的任务。Atsec Information Security公司的Sal La Pietra在幕后牵线帮忙，使得翻译项目得以顺利进行。特别要感谢Stephan Müller在安全评估期间给予的密切合作。我要诚恳地感谢此次评估所涉及的所有其他惠普和Red Hat的员工，感谢Claudio Kopper和Hans Lohr，同他们项目期间的合作很愉快。我还要感谢Wiley出版社的全体工作人员，是他们帮助我完成了本书。

本书德文版受到了读者和书评家的好评，但仍然收到了若干改进建议以及指出书中不准确之处的意见。我很高兴收到所有这些意见和建议。另外，出版社曾经对德文原版进行过调查，在此感谢那些回复过出版社的教师们。他们的一些建议对改进本书的当前版本很有价值。我同样要感谢这一版的审稿人，特别感谢张晓东博士，他对附录F.4提出了许多建议。

此外，我要对Christine Silberhorn博士表示感谢，他允许我将Max Planck Research Group的常规研究工作暂停四周，以便专心写作本书。我想博士在那段时间一定过得很宁静，因为我不上班就没人总去骚扰他，要在他的MacBook上安装Linux！

此外，我要感谢我的家人在生活的各个方面所给予的支持。对这种必不可少的帮助，我心中绝不仅仅是感激！最后，我必须感谢我的爱妻Hariet Fabritius。她对家里的作者给予了无限耐心。该作者工作起来不但日夜不分，而且经常混淆母语和C语言。很多时候在此人有点发疯（见下文）的时候，她还得把他拯救出来。既然现在又有了更多的空闲时间，我不仅盼望着美好的假期，而且还可以为她的笔记本电脑安装一个合适的操作系统！在写致谢的时候，我突然意识到，为什么看到我接近的时候，人们总是赶快把笔记本电脑锁起来了。我还真是有点儿发疯……

引　言

UNIX操作系统简单而一致，但只有天才（至少程序员）才能领会并欣赏其简单性。
——Dennis Ritchie

作者注：是的，我们疯了。预先警告：你们也会一样。
——Benny Goodheart与James Cox

UNIX操作系统以简单、一致、优雅的设计著称，这种真正非凡的特性使得UNIX系统在超过1/4世纪的时间里影响了整个世界。而且，正是由于Linux的蓬勃发展，发源于UNIX的思想才依然活力依旧，并在可预见的未来其发展势头会一直持续下去。

UNIX和Linux操作系统带有某种强烈的吸引力，前述的两段引文很好地描述了这种吸引力的精神本质。UNIX操作系统诞生于贝尔实验室，Dennis Ritchie是其发明人之一。他在引文中提到，只有天才才能欣赏UNIX操作系统的简单性，这是否是完全正确的呢？显然不是，因为Ritchie在经过全面考虑后立即改口，称程序员也同样有资格欣赏UNIX操作系统。

UNIX和Linux操作系统的源代码复杂、文档少、对程序员的要求高，要想看懂这些代码并不是一件容易事。但只要一个人开始感受到内核源代码中所能获得的远见卓识，那就很难逃脱Linux的吸引力了。在此我给读者提出一个忠告：一旦开始潜心钻研操作系统内核，就很容易沉溺于此种乐趣之中。事实上，Benny Goodheart和James Cox在其书*The Magic Garden Explained*（该书解释了UNIX System V的内部实现机制）的序言中，早已对此做过说明（前文第二段引文）。当然，Linux肯定也能让读者发疯！

本书可用作指南和手册，引导读者阅读内核源代码，并使得读者能够更敏锐地体会到这些代码的美丽、优雅，以及相关概念在设计上的美学取向。当然，要理解内核，是有一些前提条件的。读者必须熟悉C语言。如果对您来说C只是一个字母，或者是一门外语，那可以休矣。操作系统绝非仅仅是一个“开始”按钮，熟悉少量相关的算法绝对是有益无害的。最后，如果读者对计算机体系结构有一定的了解，而不是仅仅知道如何造一个新奇的机箱，那就更有用了。从学术观点来看，上述要求比较接近于系统程序设计、算法和操作系统原理课程。本书的前一版本已经在几所大学用于向高年级本科生讲授Linux原理，我希望这一版也能用于同样的目的。

本书不可能对前述的所有主题都进行详细讲解，在读者思考拿在手里的这本大部头书的时候（当然也可能因为书太厚，没拿在手里），读者肯定会同意我的看法。如果某个主题与内核没有直接的关系，但对理解内核的运作机制是必需的，那么我会在书中相关之处简要介绍它。如果读者需要更透彻地理解相关知识，可以查阅我推荐的有关计算机原理方面的图书。市面上有大量的教科书可供选择，我觉得某些图书特别具有启发性，包括Brian W. Kernighan和Denis M. Ritchie的*C Programming Language*

[KR88]；Andrew S. Tanenbaum的*Modern Operating Systems* [Tan07]（该书是关于一般操作系统的基础知识），Andrew S. Tanenbaum和Albert S. Woodhull的*Operating Systems:Design and Implementation* [TW06]（该书是关于UNIX操作系统（Minix）的），W. Richard Stevens和Stephen A. Rago的《UNIX环境高级编程（第2版）》[SR05]（该书是关于用户空间程序设计的），还有John L. Hennessy和David A. Patterson的两本书*Computer Architecture*和*Computer Organization and Design* [HP06, PH07]（这两本书是关于计算机体系结构基础的）。上述图书都是公认的经典。

此外，附录C包含了一些内核中用到的GNU C编译器扩展的相关信息，但这些扩展在一般的程序设计中并未广泛应用。

在撰写本书第一版时，内核的发布基本上不存在预定计划。正如我在附录F中讨论到的，这一点在内核2.6的开发期间发生了很大的变化，内核开发者在这方面做了很好的改进，开始以可预测的间隔周期性地发布新版本。我所讨论的内容集中于内核版本2.6.24，但也包含了一些对2.6.25和2.6.26版本的引用，这两个版本是在本书定稿后发布的，只不过发布时本书尚未出版。由于对整个内核的许多全面的修改已经合并到2.6.24版本，因此选择这个版本作为本书的目标还算是不错。虽然与本书中讨论的代码相比，在比较新版本的内核中，某些细节已经发生了变化，但大的方面会保持一段时间不会改变。

在讨论内核的各个组件和子系统时，我试图忽略不重要的细节，以避免使本书的篇幅过长。同样，我尽力保持本书的行文与内核源代码之间的联系。目前的情况还是比较幸运的，由于Linux的存在，使得我们能够查看一个真正的、可工作的、产品级操作系统的源代码，因此如果忽视了内核的这种本质性的方面，那将是可悲的。为保证书的篇幅不至于太长，我只能选择内核源代码中那些最关键的部分进行陈述。在理解Linux内核的结构和实现的过程中，阅读和使用实际的源代码是必不可少的一个步骤。附录F介绍了一些技巧，能够使得阅读和使用源代码容易一些。

关于Linux（和一般的UNIX操作系统）的一个特别有趣的事实是：它很能调动人的情绪。在因特网上有关操作系统的Flame wars（特指UseNet上的激烈争论）和热烈的技术辩论可能是一个例子，但有哪个UNIX以外的操作系统会专门有一本小册子（指*The Unix-Haters Handbook*[GWS94]，由Simson Garfinkel等编辑）来论述憎恶这种系统到底有多好呢？在为第一版写序言时，我提到，某个国际软件公司用难解的控告和争论来应对Linux，这对未来而言并不是坏信号。五年以后，形势已经改善，前述的厂商已经私下接受了下述的事实：Linux已经成为操作系统领域中一个重要的竞争者。在下一个五年，情况当然会变得更好。

毫不夸张，我承认自己肯定是被Linux迷住了（有时候，我可以肯定自己几乎因此而疯狂）。如果本书能够感染到你，那么我为写作此书付出的大量心血都是值得的！

改进建议和批评意见可以发送到wm@linux-kernel.net，或经由www.wrox.com反馈给我。当然，如果有人告诉我他很喜欢这本书，那我会非常高兴！

本书涵盖的内容

本书讨论了Linux内核的概念、结构和实现。各章分别介绍了下述主题。

- 第1章概述Linux内核，讲述了内核的总体图景，后续章节则根据总体结构对内核进行更详细的研究。
- 第2章讨论了多任务、调度和进程管理的基本知识，并分析了这些基本技术和概念抽象的实现方式。
- 第3章讨论了如何管理物理内存。本章既讨论了内核与相关硬件的交互，也讨论了内核内部通

过伙伴系统和slab分配器来分配内存的方式。

- 第4章继续对内存进行讨论，讲解了用户空间的进程如何访问虚拟内存，以及在内核层面实现虚拟内存视图所需要的详细的数据结构和相关机制。
- 第5章介绍了保证内核能够在多处理器系统上正确运作所需的机制。此外，本章还介绍了进程如何相互通信。
- 第6章引导读者理解如何编写设备驱动程序，使内核支持新的硬件。
- 第7章阐述了模块机制，该机制能够向内核动态添加新的功能。
- 第8章讨论了虚拟文件系统，这是内核中一个一般的间接层，能够支持各种各样的不同文件系统，包括物理文件系统和虚拟文件系统。
- 第9章讲解了Ext文件系统族，包括Ext2和Ext3文件系统，这是很多Linux系统安装的标准选项。
- 第10章继续讨论文件系统，包括procfs和sysfs。这两个文件系统并非用来存储信息，而是向用户层提供关于内核的元信息。此外，本章阐述了一些减轻编写文件系统负担的方法。
- 第11章给出了Ext文件系统属性和访问控制表的实现方式，这两者有助于提高系统的安全性。
- 第12章讨论内核中网络的实现，内容集中于IPv4、TCP、UDP和netfilter。
- 第13章介绍了系统调用的实现方式，系统调用是从用户层请求内核服务的标准机制。
- 第14章对中断触发内核活动的方式进行了分析，并介绍了内核中将工作延迟至后续时间点执行的机制。
- 第15章说明了内核对时间相关功能的处理，包括了高低两种分辨率的情形。
- 第16章讨论了借助于页缓存和块缓存来加速内核操作。
- 第17章讨论了如何对内存中缓存的数据与持久存储设备上的数据源进行同步。
- 第18章介绍了页面回收和页交换的相关机制。
- 第19章介绍了审计的实现，审计负责详细记录内核的活动。
- 附录A讨论了内核所支持的各种计算机体系结构的特点。
- 附录B简述了有效使用内核源代码的各种工具和方法。
- 附录C提供了关于C语言的一些技术札记，并讨论了GNU C编译器的结构。
- 附录D给出了内核的启动过程。
- 附录E介绍了ELF二进制格式。
- 附录F讨论了内核开发的许多社会性的方面，以及Linux内核社区。

目　录

第1章 简介和概述

操作系统不仅是信息技术中非常吸引人的一部分，而且还是公众争论的主题[①]。在此发展过程中，Linux发挥了举足轻重的作用。然而仅仅10年前，学术用操作系统和商用操作系统还是有着严格区分的：前者相对简单而且可获得源代码；对后者而言，虽然不同的操作系统性能各不相同，但其源代码一直都是受到良好保护的秘密。现在，任何人都可以从因特网下载Linux（或任何其他自由操作系统）的源代码进行研究。

Linux现在已经安装到了数百万台电脑上，无论是家庭用户还是专业人员，都可以在Linux上执行各种任务。无论是手表中的微型嵌入式系统，还是大规模并行大型机，Linux都可以在无数领域大展身手。而这使得Linux的源代码非常有趣。一个合理可靠、基础牢固的概念（UNIX操作系统）结合了强大的创新以及学术性操作系统所缺乏的解决问题的强烈倾向，这就是为什么Linux具备如此强大吸引力的原因。

本书描述了内核的主要功能，解释了其内部的结构，并研究了其实现。由于所讨论主题的复杂性，我假定读者已经对操作系统和C语言系统程序设计有一定的基础（当然，对Linux系统的熟悉是不言而喻的）。我会简要介绍与常见操作系统问题相关的几个基础概念，但本书主要的内容则集中于Linux内核的实现。市场上有许多讲述操作系统基础概念的教材，对某一特定主题不熟悉的读者，可以找一本看看。例如，Tanenbaum写的两本杰出的入门书籍（[TW06]和[Tan07]）。

本书要求读者有牢固的C语言程序设计基础。因为内核使用了C语言的许多高级技巧，尤其是GNU C编译器的许多专门特性。附录C讨论了C语言的一些精微之处，即使优秀的程序员可能也未必熟悉这些。由于Linux必然与系统硬件（特别是CPU）有非常直接的交互，因此了解一点计算机结构的基础知识是很有用的。该主题也有很多入门书籍可用，在参考文献章节中列出了一些相关书籍。在深入讲解CPU的知识时（大多数情况下，我都以IA-32或AMD64体系结构为例，因为Linux在这些体系结构上很常用），我会解释相关硬件的细节。在讨论不常见的机制时，我会解释机制背后的一般性概念，但对于某个特定的特性如何在用户空间中使用，则需要读者查询书中指明的手册页。

本章将概述内核所涉及的各种领域，并在后续章节中对相应的子系统进行长篇阐述之前，先行说明其基本关系。

---

① 本书不打算参与意识形态上的讨论，诸如Linux是否是一个真正的操作系统。当然它事实上只是一个内核，没有其他组件是无法正常运转的。在我谈到Linux操作系统而没有明确地提及类似工程的简称时，这并不意味着我没有意识到该工程的重要性。这里的类似工程主要是指GNU工程，如果在名称上使用Linux而不是GNU/Linux，该工程的人士一般会很敏感。我的理由简单而实用。我们需要建立何种界限，才能在引用时不产生像GNU/IBM/RedHat/HP/KDE/Linux这样冗长的结构呢？如果读者觉得这个脚注没有意义，可以参考网页www.gnu.org/gnu/linux-and-gnu.html，该文总述了GNU工程的地位。在澄清了所有的意识形态问题之后，我保证在本书其余的部分不会再出现长达半页的脚注了。

由于内核的演变比较快速，读者很自然会问本书内容涵盖了哪一个内核版本。我选择了2.6.24版本的内核，该版本发布于2008年1月末。内核开发的动态性意味着，在阅读本书时，新版本的内核应该已经发布，所以某些细节很自然会有所改变，这是不可避免的。如果不是这样，那Linux将会成为一个死气沉沉、毫无乐趣的系统，读者也很可能就不会选择本书了。尽管一些细节将会发生变化，但书中描述的概念在本质上是不变的。对于2.6.24版本来说，这一点特别正确。因为与更早的版本比较，该版本有一些根本性的改动。很自然，开发者也无法隔一夜就折腾一些此类特性出来。

1.1　内核的任务

在纯技术层面上，内核是硬件与软件之间的一个中间层。其作用是将应用程序的请求传递给硬件，并充当底层驱动程序，对系统中的各种设备和组件进行寻址。尽管如此，仍然可以从其他一些有趣的视角对内核进行研究。

- 从应用程序的视角来看，内核可以被认为是一台*增强的计算机*，将计算机抽象到一个高层次上。例如，在内核寻址硬盘时，它必须确定使用哪个路径来从磁盘向内存复制数据，数据的位置，经由哪个路径向磁盘发送哪一条命令，等等。另一方面，应用程序只需发出传输数据的命令。实际的工作如何完成与应用程序是不相干的，因为内核抽象了相关的细节。应用程序与硬件本身没有联系[①]，只与内核有联系，内核是应用程序所知道的层次结构中的最底层，因此内核是一台增强的计算机。
- 当若干程序在同一系统中并发运行时，也可以将内核视为*资源管理程序*。在这种情况下，内核负责将可用共享资源（包括CPU时间、磁盘空间、网络连接等）分配到各个系统进程，同时还需要保证系统的完整性。
- 另一种研究内核的视角是将内核视为*库*，其提供了一组面向系统的命令。通常，*系统调用*用于向计算机发送请求。借助于C标准库，系统调用对于应用程序就像是普通函数一样，其调用方式与其他函数相同。

1.2　实现策略

当前，在操作系统实现方面，有以下两种主要的范型。

(1) **微内核**：这种范型中，只有最基本的功能直接由中央内核（即微内核）实现。所有其他的功能都委托给一些独立进程，这些进程通过明确定义的通信接口与中心内核通信。例如，独立进程可能负责实现各种文件系统、内存管理等。（当然，与系统本身的通信需要用到最基本的内存管理功能，这是由微内核实现的。但系统调用层次上的处理则由外部的服务器进程实现。）理论上，这是一种很完美的方法，因为系统的各个部分彼此都很清楚地划分开来，同时也迫使程序员使用“清洁的”程序设计技术。这种方法的其他好处包括：动态可扩展性和在运行时切换重要组件。但由于在各个组件之间支持复杂通信需要额外的CPU时间，所以尽管微内核在各种研究领域早已经成为活跃主题，但在实用性方面进展甚微。

(2) **宏内核**：与微内核相反，宏内核是构建系统内核的传统方法。在这种方法中，内核的全部代码，包括所有子系统（如内存管理、文件系统、设备驱动程序）都打包到一个文件中。内核中的每个函数都可以访问内核中所有其他部分。如果编程时不小心，很可能会导致源代码中出现复杂的嵌套。

因为在目前，宏内核的性能仍然强于微内核，Linux仍然是依据这种范型实现的（以前亦如此）。

① CPU是一个例外情况，程序访问CPU显然是不可避免的。尽管如此，并非所有可能的指令对应用程序都是可用的。

但其中已经引进了一个重要的革新。在系统运行中，模块可以插入到内核代码中，也可以移除，这使得可以向内核动态添加功能，弥补了宏内核的一些缺陷。模块特性依赖于内核与用户层之间设计精巧的通信方法，这使得模块的热插拔和动态装载得以实现。

1.3 内核的组成部分

本节简略概述了内核的各个组成部分，以及我们将在后续章节中详细研究的各个领域。尽管Linux是整体式的宏内核，但其具有相当良好的结构。尽管如此，Linux内核各个组成部分之间的彼此交互是不可避免的。各部分会共享数据结构，而且与严格隔离的系统相比，各部分（因为性能原因）协同工作时需要更多的函数。在后续章节中，尽管我试图将向前引用的次数降至最低，但也不得不经常引用内核的其他组成部分（即其他章节）。为此我会在这里简短介绍各个组成部分，使读者能对各个部分在内核整体结构中的作用和地位有一定的印象。图1-1是一个粗略的草图，概述了组成完整Linux系统的各个层次，以及内核所包含的一些重要子系统。但要注意，各个子系统之间实际上会以各种方式进行交互，图中给出的只是其中一部分。

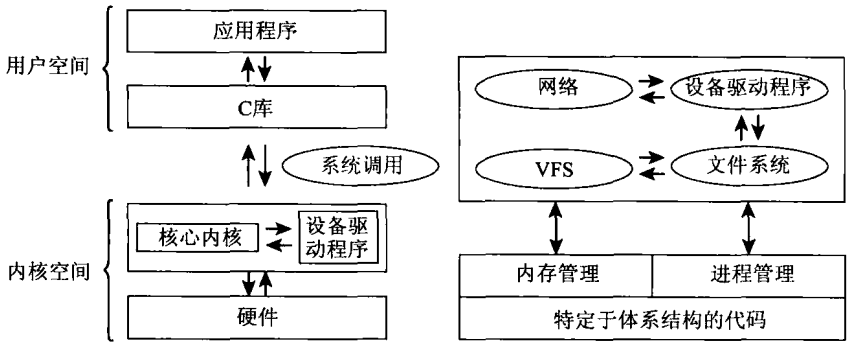


图1-1 Linux内核的高层次概述以及完整的Linux系统中的各个层次

1.3.1 进程、进程切换、调度

传统上，UNIX操作系统下运行的应用程序、服务器及其他程序都称为*进程*。每个进程都在CPU的*虚拟内存*中分配地址空间。各个进程的地址空间是完全独立的，因此进程并不会意识到彼此的存在。从进程的角度来看，它会认为自己是系统中唯一的进程。如果进程想要彼此通信（例如交换数据），那么必须使用特定的内核机制。

由于Linux是多任务系统，它支持（看上去）并发执行的若干进程。系统中同时真正在运行的进程数目最多不超过CPU数目，因此内核会按照短的时间间隔在不同的进程之间切换（用户是注意不到的），这样就造成了同时处理多进程的假象。这里有两个问题。

(1) 内核借助于CPU的帮助，负责进程切换的技术细节。必须给各个进程造成一种错觉，即CPU总是可用的。通过在撤销进程的CPU资源之前保存进程所有与状态相关的要素，并将进程置于空闲状态，即可达到这一目的。在重新激活进程时，则将保存的状态原样恢复。进程之间的切换称之为*进程切换*。

(2) 内核还必须确定如何在现存进程之间共享CPU时间。重要进程得到的CPU时间多一点，次要进程得到的少一点。确定哪个进程运行多长时间的过程称为*调度*。