Linux 内核源码剖析

—— TCP/IP 实现

上 册

樊东东　莫 澜　编著

机械工业出版社
CHINA MACHINE PRESS

Linux 内核源码剖析

——TCP/IP 实现

上册

樊东东　莫　澜　编著

机械工业出版社

本书详细论述了 Linux 内核 2.6.20 版本中 TCP/IP 的实现。书中给出了大量的源代码，通过对源代码的详细注释，帮助读者掌握 TCP/IP 的实现。本书根据协议栈层次，从驱动层逐步论述到传输层，包括驱动的实现、接口层的输入输出、IP 层的输入输出以及 IP 选项的处理、邻居子系统、路由、套接口及传输层等内容，全书基本涵盖了网络体系架构全部的知识点。特别是 TCP，包括 TCP 连接的建立和终止、输入与输出，以及拥塞控制的实现。

本书适用于熟悉 Linux 的基本使用方法，对 Linux 内核工作原理以及网络知识有一定的了解，而又极想更深入理解各个机制在 Linux 中的具体实现的用户，包括应用程序员和嵌入式程序员，以及网络管理员等。相关专业的科研人员在工作中遇到问题时，也可以查阅本书，理解相关内核部分的实现。此外，计算机相关专业的本科高年级学生和研究生，在学习相关课程（如操作系统、计算机网络等）时，可将本书作为辅助教程，与理论相结合以便更好地理解相应的知识点。

图书在版编目（CIP）数据

Linux 内核源码剖析：TCP/IP 实现/ 樊东东，莫澜编著. —北京：机械工业出版社，2010.12
ISBN 978-7-111-32373-0

Ⅰ. ①L… Ⅱ. ①樊…②莫… Ⅲ. ①Linux 操作系统—机器代码程序—程序分析②计算机网络—通信协议 Ⅳ. ①TP316.89②TN915.04

中国版本图书馆 CIP 数据核字（2010）第 212455 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策划编辑：车 忱
责任编辑：车 忱
责任印制：乔 宇
三河市宏达印刷有限公司印刷
2011 年 1 月第 1 版 · 第 1 次印刷
184mm × 260mm · 67.75 印张 · 1677 千字
0001 - 3000 册
标准书号：ISBN 978-7-111-32373-0
定价：142.00 元（上、下册）

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
社服务中心：（010）88361066
销 售 一 部：（010）68326294
销 售 二 部：（010）88379649
读者服务部：（010）68993821

网络服务
门户网：http://www.cmpbook.com
教材网：http://www.cmpedu.com

前　言

有人宣称 Linux 人才是未来 20 年 IT 职场中的王者。无论这种说法有多夸张，有一个事实是不可否认的，那就是，近年来 Linux 的市场份额不断增长，Linux 正在受到越来越多的关注乃至推崇。由于 Linux 可广泛地应用到各种系统，包括很多嵌入式系统上，以及其他的诸多优点，如开放、高效、丰富的网络功能等，这种趋势在可预计的未来还将持续。

目前，国内对 Linux 各方面的研究工作没有国外那样广泛和深入，相关的出版物水准参差不齐。特别是在网络的实现方面，有些著作针对性不强，有些则缺少了重要的传输层协议实现的论述，还有一些虽然有比较全面的介绍，却不够深入，选用的 Linux 版本也比较旧。

针对以上情况，本书选择了较新的 2.6.20 版本内核 TCP/IP 实现作详细的论述，在重要的细节处甚至逐行分析，并在此基础上，对代码背后的机制和原理作了深入的阐述，将各关键点连成一个整体，帮助读者理清整个 Linux 网络部分的脉络。作者本着严谨的态度，在写作过程中参阅了大量的中英文资料及相关的文档。相信本书可以成为那些希望深入了解 Linux 的 TCP/IP 协议栈、网络部分实现的人们的有力工具。

本书共有 33 章，通过自底向上的方法来论述 TCP/IP 的实现，从数据链路层开始，然后是网络层（IP、ICMP、IGMP、路由以及邻居子系统和 IP 组播），接下来是套接口，最后是传输层（TCP 和 UDP）。在学习的时候也可以采用自顶向下的方法（从传输层开始向下），或者结合以上两种方法。要理解一个系统的运行机制，对于一个专业人员来说，代码是最为直接也最为可靠的资源。Linux 普及面不断扩展，越来越多的人会想通过研读内核代码来了解 Linux 系统，一来是更好地解决具体工作中的相关问题，二来是从 Linux 这个高质量的操作系统中学习到更多的编程、架构等技术。本书的特点如下：

- 选择的内核版本新，内容不会在短期内过时。
- 对代码作了详细的论述，在此基础上进一步分析了代码背后的机制和原理，为读者理清了整个框架的脉络，帮助读者避免迷失在细节中。
- 写作过程中参阅了大量中英文资料和相关的文档，对内核代码更是作了长时间深入细致的研究和分析，在细节处反复推敲，以确保本书的质量。
- 书中有大量的图表，用来帮助读者直观地理解各个数据结构之间的联系及各个函数的调用关系等。

Linux 内核中的网络模块虽然是一个比较独立的模块，但仍会涉及内核中一些常用的技术和算法。因此读者在阅读本书之前需要掌握一些基础知识，对 TCP/IP 应有一定的了解。最好能够深入地阅读过 TCP/IP 相关的 RFC，了解 Linux 内核的运行机制，包括进程管理、内存管理、文件系统、系统调用、netlink、内核中多种锁（自旋锁、读写锁、RCU）等。当然在学习过程中，免不了要做些实验，因此也需要熟悉 Linux 提供的多种工具，包

括 ip、ifconfig 等命令。

本书使用的 Linux 内核版本为 2.6.20。也许读者想知道为什么选择这个版本。实际上在作者开始分析 TCP/IP 代码时，该版本是最新版本。只是因为 Linux 内核的更新很快，经过两年多的分析后，最新版本已经是 2.6.3x 了。即便如此，如果读者能够理解 2.6.20 版本的代码，则再读最新版本的代码时，应该不会太难。

由于本书内容繁多，漏洞和不足之处难免，请读者谅解并提出修改建议。作者的博客是 http://blog.chinaunix.net/u3/112243/。

在此，还要特别感谢网友衫秋南对本书提出宝贵的意见和建议。

上册目录

第1章 预备知识

随着 Linux 的蓬勃发展和普及，想深入了解内核实现的技术人员也越来越多。而要真正深入了解内核，就需要阅读和分析内核源码。

Linux 内核源码可以从包括网络在内的很多途径得到，例如从 http://www.kernel.org 下载。当然，在很多发行版中，/usr/src/linux 目录也存放内核源码。目前最新的稳定版是 2.6.36.2。

很多技术人员畏惧阅读 Linux 内核源码，因为这样庞大而复杂的系统代码，阅读起来确实有很多困难。由于操作系统由多个模块组成，包括进程管理、内存管理、文件系统、驱动程序、网络等，而 Linux 内核也采用了面向对象技术进行设计和编码，因此实际上阅读源码也没有想象的那么高不可攀。只要有恒心，不必担心水平不够，困难都是可以克服的，很多知识可以边阅读源码边学习。

做任何事情都需要有方法和工具，同样，阅读 Linux 内核源码也是如此。由于内核源码非常庞大，因此不能全面铺开，而是要按照模块一个一个去攻克，本书的目的就是指导和帮助读者学习网络模块。要想比较顺利地阅读内核网络源码，事先最好对源码的知识背景有一定的了解。对于内核网络源码来讲，基本要求是：熟悉 C 语言，最好了解 GNU 对标准 C 的扩展；熟悉 GCC 编译器以及使用方法；熟悉操作系统的基本知识；熟悉 Linux 内核通用技术，包括内存管理、下半部、锁等；熟悉 TCP/IP 的原理。

本书讲述的代码来自 Linux-2.6.20，下载网址是 http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.20.tar.bz2。

俗话说：“工欲善其事，必先利其器”。像 Linux 内核源码这样的复杂程序，阅读时遇到的问题肯定会像一个越滚越大的雪球。例如一个函数经常进行多层次的调用，一个结构会有多个结构多层次的嵌套，而且会涉及多个其他的相关文件，对代码进行多层次跟踪后，说不定一下子还回不到原先阅读的函数或结构。因此需要一个好的阅读工具。

由于大部分技术人员和爱好者都比较熟悉 Windows 或 Linux 平台，所以在此介绍两个工具。

一个是 Windows 下的 Source Insight，它是一个共享软件，有 30 天免费期，可以从 http://www.sourcedyn.com 下载。安装和使用这个软件都非常简单，安装完成并运行后，先选择“Project”菜单下的“new”命令，新建一个工程，输入工程名，接着把欲读的源码加入（可以添加整个目录）后，软件就会分析所加的源码。分析完后，就可以进行阅读了。对于打开的阅读文件，如果想看某一变量的定义，可先把光标定位于该变量，然后点击工具条上的相应选项，该变量的定义就会显示出来，对于函数的定义与实现也可以同样操作。

另一个是 Windows 和 Linux 下的 Source-Navigator，可以从 http://sourceforge.net/projects/sourcenav/files/下载。它是 Red Hat 开发的一个 IDE，很适合用来阅读源代码——因为它能很好地解决文件定位和跳转问题，功能与 Source Insight 类似。运行 Source-Navigator 之后，该软件会建立源代码的 project 并扫描源代码文件，自动建立文件之间的索引，之后便可以进行阅读了。

读者也可在线阅读源代码，网址是 http://lxr.linux.no/#linux+v2.6.20/。

1.1 应用层配置诊断工具

在学习 Linux 网络模块实现的过程中，免不了要做些实验，来验证自己的想法或查看代码流程。Linux 提供了多种配置工具用于配置内核以及网络特性。

1.1.1 iputils

除了经常使用的 ping 外，iputils 包还包含 arping（用来生成 arp 请求），网络路由发现服务器 rdisc，以及其他一些程序（tracepath、clockdiff、tftpd 和 rarpd）。

1.1.2 net-tools

net-tools 网络工具包，最常用的就是 ifconfig，route，netstat 和 arp，还包括 hostname，nameif，plipconfig，rarp，slattach，ipmaddr，iptunnel 和 mii-tool 等工具。

1.1.3 iproute2

iproute2 是 Linux 特有的、新一代的网络配置工具包。其中 ip 是最常用的命令，它可以用来配置网络设备（ip link）、IP 地址（ip address）、路由（ip route）以及其他功能。tc 也是 IPROUTE2 工具包中常用的、用来配置流量控制的工具。除了以上两个工具，还包括 ss，nstat，ifstat，rtacct，arpd，lnstat，maketable，normal，pareto 和 paretonormal。

1.2 内核空间与用户空间的接口

内核提供接口给用户空间程序，以便用户进程进行信息的读取或配置。除了经典的系统调用接口，Linux 还可通过几个特殊的接口，包括虚拟文件系统和 netlink 套接口，获取相关信息。

procfs 和 sysctl 都可以导出内核内部信息，但是 procfs 主要用于导出只读数据，而 sysctl 导出的信息是可写的（只有超级用户可写）。如果导出的是只读数据，那么选择 procfs 还是 sysctl 就与导出数据的数量有关。如果导出的是简单变量，那么应该使用 sysctl。反之，如果导出大量复杂的数据，并且要求导出的数据有固定格式，就应该使用 procfs。

1.2.1 procfs

procfs 是一个虚拟文件系统，通常挂接在/proc 目录下，内核通过文件的形式将内部信息展现给用户空间程序。这些文件都不是磁盘文件，但是可以使用 cat 命令读取它们，或者通过重定向符号（>）写它们，甚至可以像普通文件一样设定它们的读写权限。

网络代码注册的文件一般位于/proc/net 目录下，在该目录下存在不少文件，有一种比较特殊的文件，比如 tcp、udp 等。这种文件的格式比较固定，Linux 称之为综合文件（synthetic files），它们最大的特点就是由一系列记录组成，类似于数据库表中的一条条记录。用这种格式来描述系统中的一些统计或状态是比较适合的，因此 proc 文件系统特地增加了对这种文件类型的支持。因为它只存在于 proc 文件系统中，文件系统提供了对此文件框架的支持，而在使用此类型文件时，只需关心数据的处理即可。

大多数网络功能在初始化时，都会在/proc/net 中注册一个或多个这样的文件，无论初始化

动作发生在系统启动时还是模块加载时。当用户读取某个文件时，内核会调用一组内核函数来输出相应的信息。

创建和删除文件可以分别调用 proc_net_fops_create()和 proc_net_remove()，这两个函数分别包装了函数 create_proc_entry()和 remove_proc_enty()。需要注意的是 proc_net_fops_create()在创建文件的同时也初始化了这个文件的操作函数。来看下面的例子。

这是 IP 组播在/proc/net 目录下注册 ip_mr_vif 文件的例子：

```
1717 static struct file_operations ipmr_vif_fops = {
1718     .owner    = THIS_MODULE,
1719     .open     = ipmr_vif_open,
1720     .read     = seq_read,
1721     .llseek   = seq_lseek,
1722     .release  = seq_release_private,
1723 };

1899 void __init ip_mr_init(void)
1900 {
....
1908 #ifdef CONFIG_PROC_FS
1909     proc_net_fops_create("ip_mr_vif", 0, &ipmr_vif_fops);
....
1911 #endif
1912 }
```

proc_net_fops_create 的三个参数所代表的含义是：文件名是 ip_mr_vif，文件属性是只读，文件的操作函数句柄是 ipmr_vif_fops。当用户读文件时，file_operations 结构里面的函数会以数据块的方式将数据返回给用户。这在读取大量相同类型数据的时候很有用。例如，可以一次读取组播虚拟接口。

ipmr_vif_open()会进行一系列初始化：它会注册一组函数指针，这些函数指针在 procfs 的例程中被调用，用于遍历返回给用户的数据：一个函数用于初始化打印动作，另一个用于向前移动一次指针，还有一个用于打印一个数据项。这些函数会在内部保存必要的上下文信息以便了解当前的打印项在哪里，以及可以从哪个地方重新开始打印等。

```
1686 static struct seq_operations ipmr_vif_seq_ops = {
1687     .start = ipmr_vif_seq_start,
1688     .next  = ipmr_vif_seq_next,
1689     .stop  = ipmr_vif_seq_stop,
1690     .show  = ipmr_vif_seq_show,
1691 };

1693 static int ipmr_vif_open(struct inode *inode, struct file *file)
1694 {
....
1702     rc = seq_open(file, &ipmr_vif_seq_ops);
1703     if (rc)
1704         goto out_kfree;
....
1715 }
```

1.2.2 sysctl(/proc/sys 目录)

sysctl 接口允许用户读取或者修改内核参数。在用户空间可以通过两种方法访问 sysctl 导出的变量。一种方法是通过 sysctl 系统调用，另一种是通过 procfs。如果内核支持 procfs，它会在/proc 目录下增加一个特殊的目录（/proc/sys），这个目录里包含了 sysctl 导出变量的列表。

procps 包里的 sysctl 命令可以用于配置 sysctl 接口导出的内核参数，实际上这个命令是通过/prco/sys 目录下的文件与内核通信的。大多数 Linux 发行版的内核都默认包含了 sysctl 的支持。

通过 sysctl 系统调用和命令来设置和获得运行时内核的配置参数是一种有效的方式，通过这种方式，用户可以在内核运行的任何时刻修改和获取内核的配置参数。例如，通过 cat /proc/sys/net/ipv4/ip_forward 来获取内核网络层是否允许转发 IP 数据报。通过 echo 1 > /proc/sys/net/ipv4/ip_forward 将内核网络层设置为允许转发 IP 数据报。通过 cat /proc/sys/kernel/ostype 可以获取操作系统的类型，其实也可以通过 sysctl 系统调用达到同样的效果。

```
#include <unistd.h>
#include <sys/syscall.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <linux/sysctl.h>

int _sysctl(struct __sysctl_args *args );

#define OSNAMESZ 100

int main(void)
{
    struct __sysctl_args args;
    char osname[OSNAMESZ];
    size_t osnamelth;
    int name[] = { CTL_KERN, KERN_OSTYPE };

    memset(&args, 0, sizeof(struct __sysctl_args));
    args.name = name;
    args.nlen = sizeof(name)/sizeof(name[0]);
    args.oldval = osname;
    args.oldlenp = &osnamelth;

    osnamelth = sizeof(osname);

    if (syscall(SYS__sysctl, &args) == -1) {
        perror("_sysctl");
        exit(EXIT_FAILURE);
    }
    printf("This machine is running %*s\n", osnamelth, osname);
    exit(EXIT_SUCCESS);
}
```

图 1-1 显示了以 root_table 为顶点形成的树状结构。

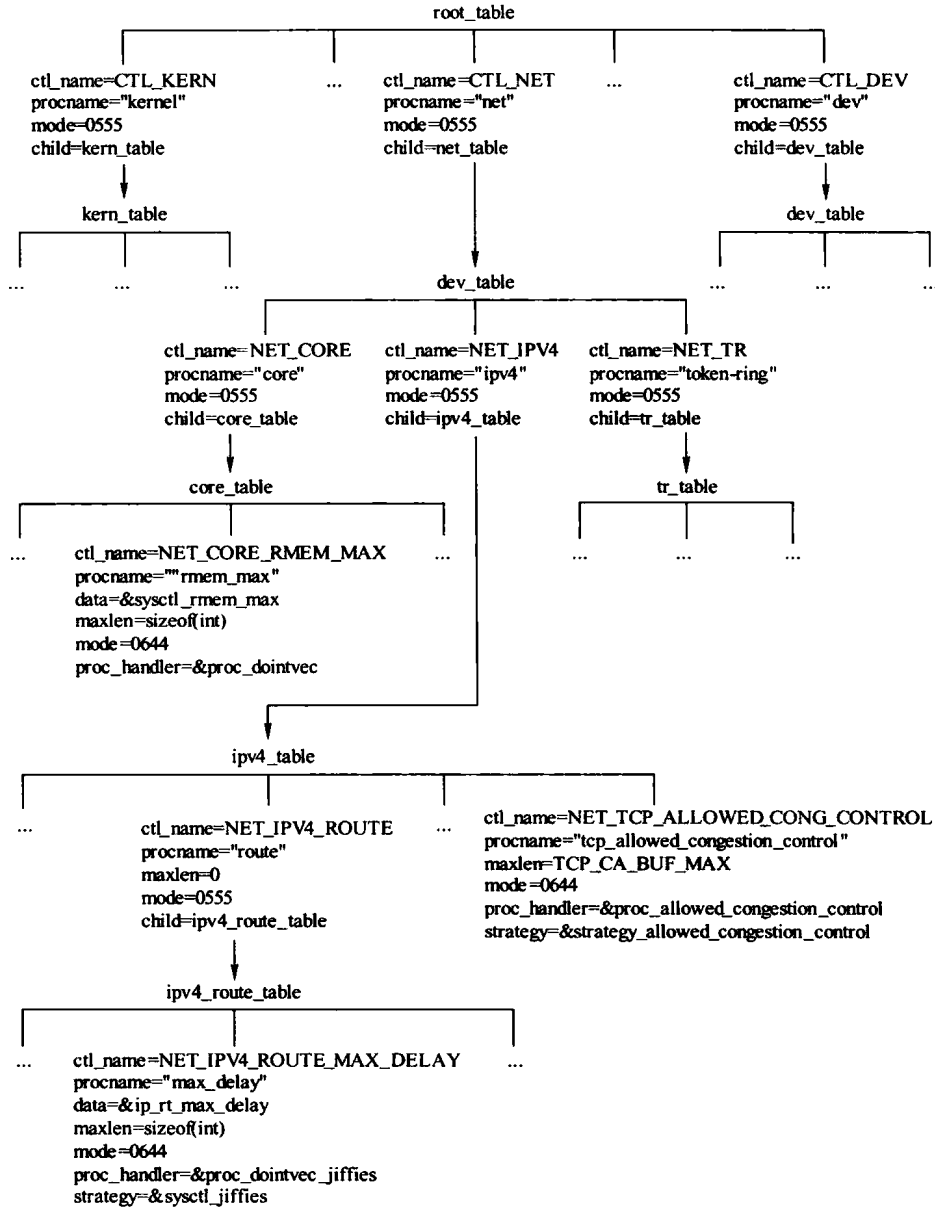

图 1-1 sysctl 的组织

1.2.3 sysfs(/sys 文件系统)

sysfs 是 Linux 2.6 提供的一种虚拟文件系统，这个文件系统不仅可以把设备和驱动程序的信息从内核空间导出到用户空间，也可以对设备和驱动进行配置。

sysfs 的目的是把一些原本在 procfs 中的设备独立出来，以设备树的形式呈现给用户。sysfs 最初名称为“driverfs”，当初只是为了要对新的驱动程序模型除错而开发出来的，然而随着代码的更新，新的“驱动程序模型”和“driverfs”证明了对内核中的其他子系统也有用处。Kobjects 开发出来之后，用于核心部件的管理机制，因此 driverfs 也被更名为 sysfs。

1.2.4 ioctl 系统调用

ioctl 系统调用可以操作一个文件，它通常用于实现特殊设备的操作，而这些操作在标准的文件系统调用中没有提供。ioctl 也可以操作套接口描述符，net-tools 工具包就是通过 ioctl 系统调用与内核交互的。

1.2.5 netlink 套接口

netlink 套接口是网络应用程序与内核通信最新、最常用的接口，IPROUTE2 包中的大多数命令都使用这个接口。netlink 套接口的描述在 RFC 3549 中。

netlink 套接口使用起来非常简单，通过套接口标准的 API 来打开、关闭，或者发送和接收信息。要在用户态创建一个 netlink 套接口，代码如下所示：

```
socket(PF_NETLINK, SOCK_DGRAM, NETLINK_ROUTE)
```

在内核中定义了多种协议，每一种都被协议栈中的一个或一组内核组件使用。例如上述代码中的 NETLINK_ROUTE 协议就会被多个模块使用，如路由和邻居协议，而 NETLINK_FIREWALL 用于防火墙模块。

1.3 网络 I/O 加速

尽管技术有了巨大的进步，但是 TCP/IP 协议栈的处理方式却几乎没有变化。也就是说，即便用户使用最先进的 CPU，依然要处理那些未经优化的 TCP/IP 协议，由此产生巨大的系统开销。例如，TCP/IP 的传输过程中需要封装、解包，这些动作对于处理器而言并不是一个复杂的过程，但是会占用处理器周期，而且网络带宽越高，这个问题越严重。系统开销的增大不仅仅表现在占用较多的处理器周期，还会导致处理网络相关数据时的内存访问效率降低。这又会进一步降低 CPU 效能和网络效率。

过去，网络流量较低，处理网络相关数据所产生的开销，远远低于用于执行正常任务的开销，所以并未引起重视。现在，随着网络流量大幅度提升，处理网络相关数据所产生的系统开销越来越不能忽视，甚至已经影响到了正常应用。现有几种解决方案：

（1）TSO（TCP Segmentation Offload）

通过网络设备上的专用处理器处理部分或者全部的封包，借此来降低对于系统处理器资源的占用，不过这种解决方案只对具有某些特征的数据包有效。

（2）RDMA（Remote Direct Memory Access，远程直接内存访问）

发送端系统直接将有效数据送至目的系统指定的内存中，无需移动数据包的时间消耗，因此大大提升了网络传输的效率。但是这种技术需要专用的网络设备，应用程序也需要进行修改，甚至还增加了一个 RDMA 层的封装过程，而且这种操作风险较高，因此目前看来还不是一个吸引人的解决方案。

（3）Onloading 技术

将系统处理器作为处理网络流量的第一引擎，尽可能地提升 CPU 处理网络数据包的效率，这种思想已经被英特尔借鉴。

目前，Linux 已经支持 TSO 和英特尔 I/O 加速技术，对于网络设备的 DMA 本书不作论述。要支持英特尔 I/O 加速选项 CONFIG_INTEL_IOATDMA，必须在编译前选择支持“Intel I/OAT