LINUX

Linux网络编程

林宇　郭凌云　编著

人民邮电出版社
www.pptph.com.cn

Linux 网络编程

林　宇　郭凌云　编著

人民邮电出版社

图书在版编目（CIP）数据

Linux 网络编程/林宇，郭凌云编著，—北京：人民邮电出版社，2000.11
ISBN 7-115-08766-0

I.L… II.①林…②郭… III.操作系统（软件），
Linux-计算机网络-程序设计 IV.TP316.89

中国版本图书馆 CIP 数据核字（2000）第 46332 号

内 容 提 要

本书比较完整地介绍了 Linux 网络编程的知识。全书共分成四篇：基础知识篇、初级应用篇、应用提高篇和高级编程篇。在第一篇（第一至三章）中，介绍了 Linux 最基本的概念：文件系统和进程系统，对这两个概念进行了比较深入的说明和分析。在第二篇（第四至八章）中，主要讲述网络应用的基础，介绍了基本套接字编程、输入/输出的基本模型、带外数据的发送和接收以及服务器编程的模型、TCP/UDP 编程比较等。在第三篇（第九至十二章）中，主要讲述如何构造网络应用，介绍了高级套接字函数的使用及如何编写守护进程、如何传递复杂的数据结构、如何编写 RPC 应用。在第四篇（第十三至十六章）中，侧重于服务器性能的提高，介绍了服务器的预创建技术、使用 UNIX 套接字来实现父进程对子进程的动态管理、多线程编程，并在最后列举了一个较完整的网络应用实例。

本书主要读者对象为网络编程人员，对于网络编程的初学者可以阅读全书，对于具有一定网络编程经验的读者可以有选择地阅读本书相关章节。

Linux 网络编程

◆ 编　　著　林　宇　郭凌云
责任编辑　梁　凝
◆ 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@pptph.com.cn
网址　http://www.pptph.com.cn
北京汉魂图文设计有限公司制作
北京朝阳展望印刷厂印刷
新华书店总店北京发行所经销
◆ 开本：787×1092　1/16
印张：31.75
字数：798 千字　　2000 年 10 月第 1 版
印数：1－5 000 册　　2000 年 10 月北京第 1 次印刷
ISBN 7-115-08766-0/TP·1812

定价：46.00 元

前　言

在 Windows 98/NT 操作系统普遍运用在 PC 上的今天，Linux 系统异军突起，成为 Windows NT 在服务器方面的最有力的竞争者。由于 Linux 出色的性能和相对较低的价格，使其已经成为中低档服务器的首选操作系统。

Linux 是 PC 硬件平台上的类 UNIX 操作系统，它继承了 UNIX 强大的功能和极佳的稳定性，并降低了对硬件环境的要求。但是 Linux 对于普通用户来说使用较为困难，系统维护的工作需要较深入的操作系统的相关知识。随着 Linux 本身的不断发展，Linux 将会呈现出越来越友好的用户界面。

在网络中已出现了许多研究、探讨、普及 Linux 的站点，但是这些站点上的内容比较零散，很不系统，并不适合 Linux 的学习者。

本书力图比较完整地介绍 Linux 操作系统的知识和网络编程的知识，本书分成四篇：基础知识篇、初级应用篇、应用提高篇和高级编程篇。第一篇主要介绍 Linux 的最基本的概念：文件系统和进程系统，对这两个概念进行了比较深入的说明和分析，希望读者在进入网络编程之前，首先对操作系统的一些概念有比较清晰的认识。第二篇主要讲述网络应用的基础，介绍了基本套接字编程、输入/输出的基本模型、带外数据的发送和接收以及服务器编程的模型、TCP/UDP 编程比较等。第三篇主要讲述如何构造网络应用，介绍了高级套接字函数的使用及如何编写守护进程、如何传递复杂的数据结构、如何编写 RPC 应用。第四篇侧重于服务器性能的提高，介绍了服务器的预创建技术、使用 UNIX 套接字来实现父进程对子进程的动态管理、多线程编程，并在最后例举了一个较完整的网络应用实例。

本书力图做到概念和思路清晰，并对相应的概念提供了大量的编程实例。读者可以在本书介绍的知识的基础上开发出更加复杂的网络应用程序。

本书由林宇、郭凌云等人编著。在编写过程中，高波对本书的策划和总体结构提了许多诚恳的建议，在此深表谢意。

本书基本上覆盖了网络编程的内容，但是限于本书的篇幅，还有一些内容不能完全叙述。比如在通信协议中，有限状态机是经常使用的，但由于本书毕竟不是网络方面的专著，因此不可能做到面面俱到。限于时间和作者水平，书中难免有所纰漏，请读者批评指正。

作者

2000 年 6 月

目 录

第一篇

基础知识篇

本篇内容包括 3 章，主要讲述了 Linux 操作系统和网络编程的基础知识，具体内容如下：

在第一章中，我们将介绍 Linux/UNIX 操作系统的两个基本的概念：文件系统和进程。操作系统的其他概念都是从它们引申而来。对于文件系统，我们讲述了文件系统的总体结构、虚拟文件系统转化、文件和目录文件的结构、特殊文件、文件的权限等问题。对于进程，我们讲述了进程的概念、进程的核心数据结构、进程状态的转化、进程控制的原理（进程创建的原理、进程 exec 的原理、进程 wait 的原理等）。

在第二章中，我们将说明进程间通信的基本方法：信号、管道、消息队列、共享内存、信号量组等。进程间通信是网络编程的重要知识，因为 Linux 操作系统是多用户多任务的操作系统，进程间同步和通信是必须很好理解和掌握的内容。

在第三章中，我们介绍与网络协议相关的基本概念：网络的层次性、协议和服务的概念、网络层次间数据的传递过程、TCP/IP 体系、域名系统、TCP 协议、IP 数据包头部等等。

本篇的知识是后面章节的基础。对于非常熟悉 Linux 操作系统基础知识的读者，可以跳过本篇中的第一、二章。对于熟悉网络协议等概念的读者可以跳过本篇中的第三章。

第一章　文件系统和进程系统

本章将简要地介绍 Linux 操作系统的两个基本的概念：文件系统和进程系统。本章的思路是先说明操作系统的一般概念，然后讲述 Linux 中的具体实现，最后介绍编程。

本章第一节首先以 System V 文件系统为例讲述文件系统的总体结构，然后介绍超级块中两个重要的数据结构：空闲数据块表和空闲索引节点表。空闲数据块表用于数据块分配和回收，空闲索引节点表用于索引节点的分配和重用。接着说明了 Linux 中虚拟文件系统转化的原理，通过虚拟文件系统转化，Linux 可支持多种文件系统。最后介绍了文件和目录的结构，以及特殊的文件，如管道文件、设备文件等。

第二节从编程的角度介绍了文件系统的系统调用的接口和 C 库函数接口。

第三节从程序并行运行中出现的问题引出进程的概念，然后介绍操作系统中进程的物理表示和进程的虚空间、进程上下文等概念。通过 Linux 中的核心数据结构 task_struct 说明 Linux 中进程的某些相关实现，包括进程间的层次关系、进程的调度、进程上下文的切换、进程的信号等。最后说明了 fork 和 exec 的原理，并给出了它们的使用实例。

1.1　文件系统

1.1.1 文件系统的总体结构

1. 文件系统的概述

在将磁盘进行格式化后，可将物理硬盘划分成不同的分区，每个分区上可建立与之联系的文件系统。文件系统的最顶端是根文件系统，其它文件系统可被安装在根文件系统的某个节点上，从用户角度看，一个典型的 Linux 文件系统的结构如图 1.1 所示。

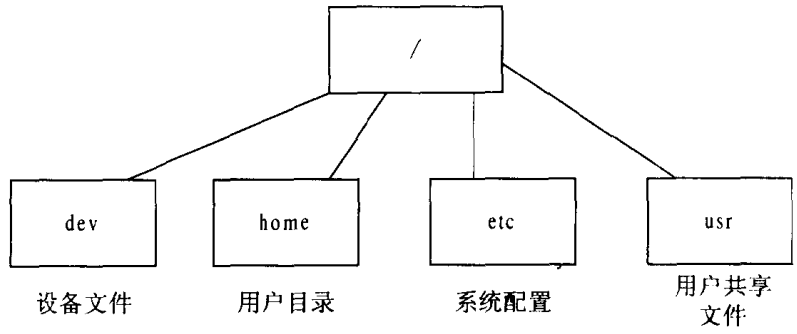

图 1.1　从用户角度看的 Linux 文件系统的结构

/dev 目录提供系统中各种设备的说明；/etc 提供系统各种关于网络、用户帐号的配置信

息；/home 是用户帐号的缺省目录；/usr/include 存放各种 include 的库文件。

从文件系统的实现角度看，可按层次分成应用程序、系统调用、文件子系统、高速缓冲、设备驱动和具体的存储设备等几个层次，如图 1.2 所示。

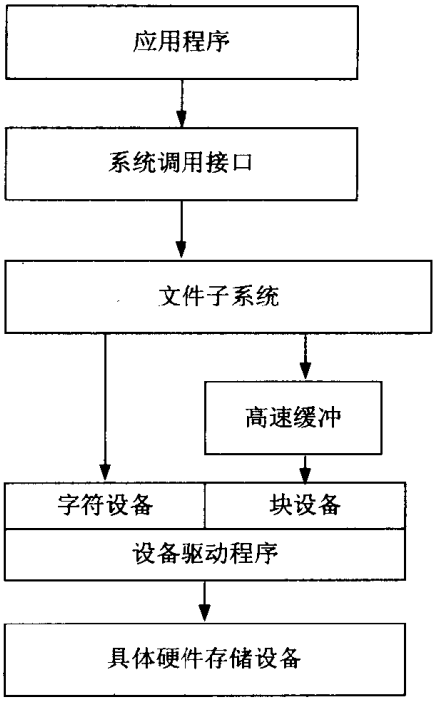

图 1.2　文件系统的层次图

应用进程通过系统调用来访问文件系统，便于隐蔽不同文件系统具体设计和实现上的差异，给应用进程一个标准的通用接口。文件子系统并不直接访问具体的物理设备，它调用设备驱动进程进行操作。对于硬盘这样的高速设备，在文件子系统和设备驱动进程间通过高速缓冲机制来提高磁盘和内存间的数据传送效率。设备驱动进程用来隐蔽具体物理设备操作的差异。

2. 根文件系统和子文件系统

文件系统分成根文件系统（root filesystem）和子文件系统，根文件系统是整个文件系统的基础，不能被卸载。子文件系统通常是以根文件系统的目录树中的某个目录的形式出现的，它可以包括作为子文件系统的硬盘、软盘、光驱、网络文件系统等。根文件系统和子文件系统有各自独立的目录结构。

3. 文件系统的总体结构

Linux 支持多种文件系统，这样便于它和许多其它的操作系统共存。Linux 支持的文件系统包括：ext、ext2、xia、minix、umsdos、msdos、vfat、/proc、System V 等。为了理解文件系统的基本概念，我们将以 System V 文件系统为例进行说明。

我们可以将文件系统想像成平板式连续的具有一定结构的存储空间，其结构如图 1.3 所示。

图 1.3　文件系统的总体结构

引导块在文件系统的开头，大小通常为一个磁盘块大小。所谓引导块，顾名思义，其中包含有引导操作系统的进程。在整个文件系统由多个文件系统构成时，只有根文件系统的引导块有效。

超级块也称管理块，存放了与整个文件系统的管理相关的信息。如文件系统的大小、索引节点表的大小、空闲空间的大小、空闲块链表的头等。超级块中的信息对文件系统是非常重要的，可以通过 ustat 系统调用来获取超级块中的某些信息。

索引节点表(index node list)由若干磁盘块组成，每个文件都对应一个称为索引节点(index node)的数据结构,这个结构中包含了文件所有者、文件的大小、文件的访问权限、存取实践等信息；索引节点还通过指向文件实际数据块的一些指针来描述文件在磁盘中的分布。用户通常是通过文件路径名来访问文件，操作系统需要经过一系列的转化将用户提供的文件路径名映射到唯一的索引节点上，再通过该索引节点读取具体数据。

数据区是文件系统实际存放数据的磁盘空间。整个文件系统是连续的地址空间，但系统对空间的访问并不是以字节为单位，而是以数据块为单位，这样的设计主要是因为硬盘的读取速率较高，它和内存间进行数据交换通常是以固定的块为单位。如果数据块较大，磁盘每将磁盘的读写头定位到所需位置一次，就可以读较多的数据，因而磁盘和内存间的数据传输效率将提高。但是从数据存储的效率来看，太大的数据块将浪费磁盘空间，比如单位数据块是 4KB，则一个大小为 5000 字节的文件将要占用 2 个数据块，其中一个数据块几乎是空闲的。因此，我们只能是根据应用的环境，选择一种较好的折衷方案。

在超级块有两个很重要的数据结构：空闲数据块表和空闲索引节点表。下面将通过这两个数据结构说明文件系统是如何分配和管理数据块和索引节点的。

4. 空闲数据块表

在文件系统的数据区建立后，数据区被划分成若干数据块，要管理这些数据块就必须将空闲的数据块的块号记录下来。数据块的数量可能很大，而超级块的空间很小，不可能将所有的数据块空闲信息都写进超级块。为了解决这个问题，一种方法就是将空闲数据块的信息记录在数据区的数据块中。但是普通的数据块是用来保存文件的，而文件可以是任意字节流，所以不可能通过检查数据块本身来确定它是否用于保存空闲数据块信息。

System V 文件系统使用连接表的方式来实现。连接表的最前面的一小部分保存在文件系统的超级块中。在磁盘分区中的数据块按顺序进行编号（包括引导块和超级块，通常引导块和超级块占用第 0# 和第 1# 块空间），每一数据块对应唯一的数据块号。在文件系统刚刚建立时，所有的数据块都应当被记录在空闲数据块表中。但是超级块中用于保存空闲数据表的空间有限，所以将表的最初一部分记录在超级块中，再分配一个数据块用于记录剩余的空闲数据块信息，并将新分配的数据块的块号填在超级块的空闲数据块表的最后一项中。如此循环，如果一个数据块不足以记录信息，再分配数据块，将新分配的块的块号放在上一数据块的最后一项中，直到所有信息都被保存。图 1.4 说明了连接表的概念。

当请求新的数据块时，总是从超级块的空闲数据表中按从小到大的顺序进行分配。当超级块中只剩下最后一项（块 350）时（记住这不是下一次要分配的数据块号，而是保存空闲数据块表的数据块的块号），将最后一项所指定的数据块的内容读入超级块中，然后将块 350 分配出去，接着再像上面那样进行分配。这个过程参见图 1.5。

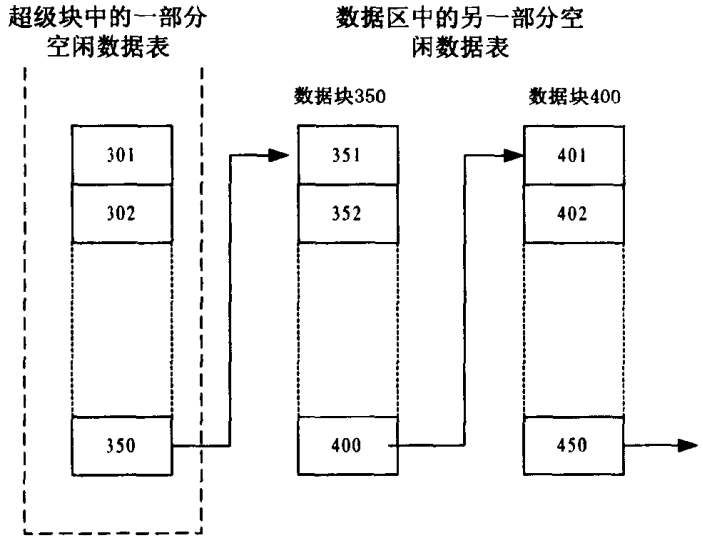

图 1.4　文件系统初始状态的空闲数据表示意图

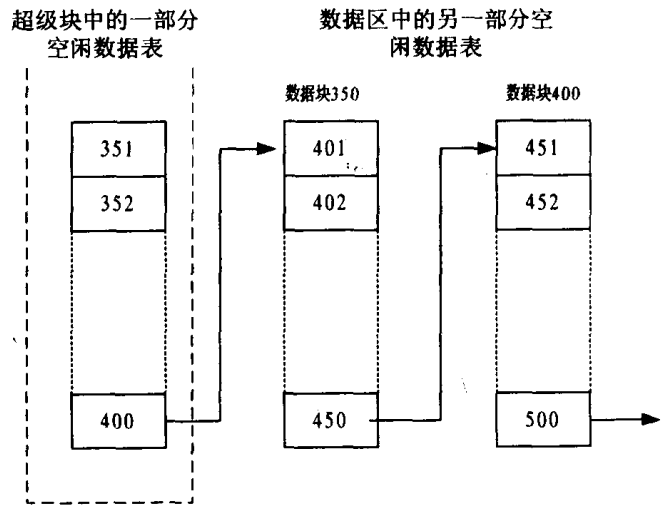

图 1.5　最初 50 块分配后的空闲数据表示意图

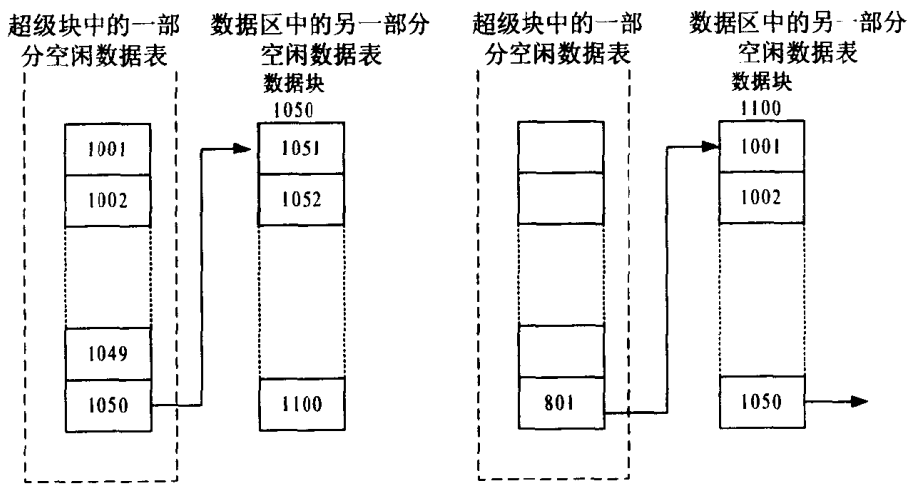

图 1.6　释放 801 块前后的空闲数据表示意图

当一些文件被删除而释放部分磁盘空间时，系统将回收这些空间。回收的过程和分配的过程相反，先将被释放的数据块号填在超级块的空闲位置。当超级块中的空闲表满时，如果又有一数据块被释放，如数据块 801，则将超级块中的空闲表写进块 801，然后将块 801 填在超级块空闲表的最后一项。这个过程参见图 1.6。

5. 空闲索引节点表

在文件系统的索引节点表存放的是文件系统能够分配的所有索引节点。索引节点表表项的数目也就是文件系统可建立的文件数量。一般情况下，索引节点表分配的频度要远小于数据块的分配频度，因为通常一个文件要占用多个数据块，而一个文件只对应唯一的索引节点。

在索引节点的结构中有一个标志，用于标记该索引节点是空闲还是被某文件使用，原则上只要将索引节点表线性地搜寻一遍，就可以实现空闲索引节点的分配。但是这样搜寻的效率可能很低，在超级块中保存了一张空闲索引节点表，当分配索引节点时，首先在超级块的空闲索引节点表中进行搜寻，这样可以大大地提高效率。但是由于超级块中的空闲索引节点表大小有限，所以其中只能保存空闲节点的很少一部分，并不是所有的空闲节点都放在空闲节点表中。当超级块中的空闲节点被分配完以后，可以再对索引节点表进行搜寻并填充，再开始新的分配。

图 1.7 显示了空闲节点表的初始状态，index 用于指示下一次请求时分配的索引节点的序号。Search_p（最后一项的内容）指示下一次从什么位置开始对索引节点表进行搜寻。由于表中的内容是对索引节点表搜寻后的结果，所以最右边的 search_p 是搜寻中找到的最大的索引节点号。

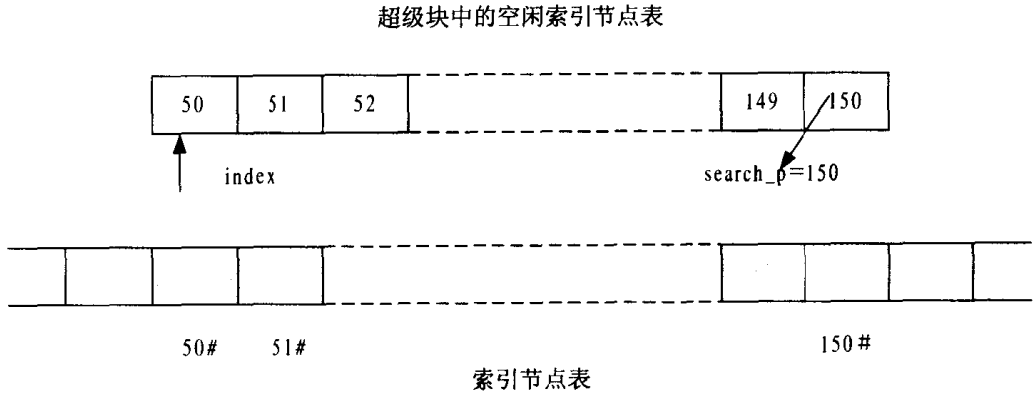

图 1.7 初始的空闲索引节点表示意图

当分配下一个索引节点时，只需将 index 向右移动，将相应的索引节点分配出去。如果超级块中的索引节点只剩下最后一项时请求到来，则从 search_p 指示的位置对索引节点表进行搜寻，因为 search_p 以前的索引节点肯定已经被分配使用了。

当一个索引节点 k 被释放时，如果空闲节点表有空位，则将其填入空位中，并调整 index 的位置。如果此时空闲节点表已经满了，则需要将 k 和 search_p 比较，如果 k<search_p，则使 search_p=k，也就是必须保证在从 search_p 开始对索引节点表进行搜寻时，search_p 之前的索引节点都是被占用的。如果 k>search_p,则不需要更新 search_p，因为 k 节点在以后需要使用时，总会被搜寻到。

6. VFS（Virtual Filesystem Switch）

Linux 通过虚拟文件系统转化（Virtual Filesystem Switch）来实现对多文件系统的支持。

Linux 的 VFS 通过在文件存储系统中有效地增加间接访问层，将对不同的文件系统的调用转化成对访问具体文件系统子程序的调用，而这些子程序的具体代码是根据具体文件系统的设计来编写的。

图 1.8 给出了 Linux 核心中虚拟文件系统和具体文件系统间的层次关系（有的书籍将这种机制直接写成为虚拟文件系统，两者的概念相近，关键是通过转化消除具体文件系统的差异，向上层提供一致的接口）。此虚拟文件系统必须能够管理在任何时刻装载到系统的不同文件系统。它通过维护一个描述整个虚拟文件系统和实际已安装文件系统的结构来完成这个工作。

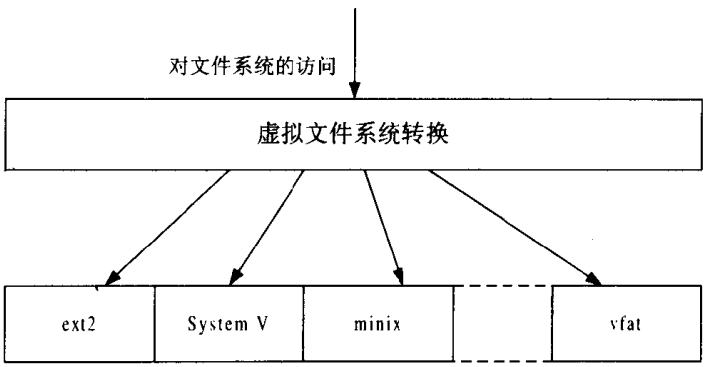

图 1.8　虚拟文件系统和具体文件系统的层次关系

VFS 使用和具体文件系统类似的方式来描述文件系统：如超级块和索引节点。VFS 索引节点描述了系统中的文件和目录以及 VFS 中的内容和拓扑结构。为了区分开虚拟文件系统和具体的文件系统的概念，我们将用 VFS 索引节点和 VFS 超级块来将它们和具体文件系统索引节点和超级块进行区分。

在系统启动和操作系统初始化时，文件系统初始化并将其自身注册到 VFS 中。这些具体文件系统可以构建在核心中，也可以设计成可加载模块。文件系统模块可以在系统需要时进行加载。装载一个基于块设备且包含根文件系统的文件系统时，VFS 必须读取其超级块。每个文件系统类型的超级块读取例程必须了解文件系统的拓扑结构并将这些信息映射到 VFS 超块结构中。

VFS 在系统中保存着一组已安装文件系统的链表及其 VFS 超级块。每个 VFS 超级块包含已安装的文件系统的信息以及一个执行特定功能的函数指针。例如表示一个已安装 EXT2 文件系统的超块包含一个指向 EXT2 相关索引节点读例程的指针。这个 EXT2 索引节点读例程像所有文件系统相关读例程一样填充了 VFS 索引节点中的域。每个 VFS 超块包含此文件系统中第一个 VFS 索引节点的指针。文件系统相关信息的映射参见图 1.9 所示，虚拟文件系统的超级块参见图 1.10。

由于系统中每个文件与目录都使用一个 VFS 索引节点来表示，系统中的进程访问目录和文件时，将通过系统调用遍历系统的 VFS 索引节点。例如键入 ls 命令，将会引起虚拟文件系统对表示此文件系统的 VFS 索引节点的搜寻，所以许多索引节点会在多次搜寻中被重复访问。与访问内存和硬盘相似，为了提高访问的效率，系统设置了索引节点 cache，读取的索引节点被保存在索引节点 cache 中以加快访问速度。如果某个索引节点不在索引节点 cache 中，则必须调用一个具体文件系统相关例程来读取此索引节点。对这个索引节点读取的结果是它将放到索引节点 cache 中以备下一次访问。不经常使用的 VFS 索引节点将会从