Linux

内核完全注释

赵 炯 编著

机械工业出版社
CHINA MACHINE PRESS

Linux 内核完全注释

赵 炯 编著

机 械 工 业 出 版 社

本书对早期的 Linux 操作系统内核（v0.11）全部源代码文件进行了详细的注释和说明，旨在让读者能够在短时间内对 Linux 的工作机理获得全面而深刻的理解，为进一步学习和研究 Linux 系统打下坚实的基础。书中首先介绍了 Linux 系统的发展历史，着重说明了各个内核版本之间的重要区别，给出了选择 0.11 版作为研究对象的原因；然后依据内核源代码的组织结构对所有代码进行了详细注释。在注释的同时，还介绍了读者应该了解的相关知识，并给出了相关的硬件信息。本书还介绍了内核源代码的组织结构及相互关系。

本书适合作为计算机专业学生学习操作系统课程的实践教材和参考书，也适合 Linux 操作系统爱好者自学，还可供具有一定基础的技术人员作为嵌入式开发应用的参考书。

图书在版编目（CIP）数据

Linux 内核完全注释/赵炯编著. —北京：机械工业出版社，2004.9
ISBN 7-111-14968-8

Ⅰ.L… Ⅱ.赵… Ⅲ. Linux 操作系统 Ⅳ. TP316.89

中国版本图书馆 CIP 数据核字（2004）第 073106 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：胡毓坚
责任编辑：车　忱
责任印制：李　妍

北京蓝海印刷有限公司印刷 • 新华书店北京发行所发行

2004 年 9 月第 1 版 • 第 1 次印刷
787mm×1092mm $^{1}/_{16}$ • 27.75 印张 • 688 千字
0001—5000 册
定价：42.00 元

凡购本图书，如有缺页、倒页、脱页，由本社发行部调换
本社购书热线电话：（010）68993821、88379646

前　言

本书是一本有关 Linux 操作系统内核工作原理的入门读物，主要目标是使用尽量少的篇幅和有限的学习时间，对完整的 Linux 内核源代码进行解剖，使读者对操作系统的基本功能和实现方式有一个全面的理解。

目前已有的阐述 Linux 内核的书籍，均尽量选用最新 Linux 内核版本进行描述，但由于这些版本的内核源代码庞大，只能对源代码进行选择性的讲解，许多实现细节被忽略。本书则对完整的 Linux 内核源代码进行了全面解剖。表面看来，本书对 Linux 早期内核版本注释的内容似乎过时，但通过学习你会发现，利用本书学习 Linux 内核，由于源代码量短小精干，因此会有极高的学习效率，能够做到事半功倍，快速入门，并为进一步选择新内核学习打下坚实的基础。

正如 Linux 系统的创始人 Linus 所说，要理解一个系统的真正运行机制，一定要阅读其源代码。系统本身是一个整体，具有很多看似不重要的细节。只有在详细阅读过完整的内核源代码之后，才会对整个系统的运作过程有深刻的理解。以后再选择较新内核源代码进行学习时，也不会碰到大问题，基本上都能顺利地理解新代码的内容。

为了帮助读者提高学习效率，作者通过对多个 Linux 内核版本进行比较和选择，最终选择了与目前 Linux 内核基本功能较为相近，又非常短小的 0.11 版作为入门学习的最佳版本。0.11 版内核源代码只有一万四千行左右(325KB)，其中包括的内容基本上都是 Linux 系统的精髓。

在阅读本书时，读者应该具备基本的有关 80x86 处理器编程和相关外围硬件的知识，还应具备使用 Linux 系统的初级技能。由于 Linus 最早是根据 M.J. Bach 的《UNIX 操作系统设计》一书的基本原理开发的，因此若能适当参考该书，则更有利于对源代码的理解。

在对每个程序进行描述时，我们首先说明程序的主要用途、输入输出参数以及与其他程序的关系，然后在程序中对代码进行详细注释。注释时对源代码和原注释不作任何改动。在代码之后是对程序中出现的一些语句或硬件方面的相关知识进行说明。

本书所需的一些基本概念均分布在各章中，这样编排主要是为了能够使读者方便地找到相关信息，而且在结合源代码阅读时，对一些基本概念能有更深的理解。另外，由于篇幅所限，我们对内核源代码多数文件开始处的版权信息作了省略，但程序中的行号仍然按原来的编号。

最后要说明的是，当你已经完全理解了本文解说的一切时，并不代表你已经成为一个 Linux 行家了，你只是刚刚踏上 Linux 的征途，具有了成为一个 Linux 高手的初步知识。这时你应该去阅读更多的源代码。

本书读者群的定位是一些知晓 Linux 系统一般使用方法或具有一定编程经验，但比较缺乏阅读目前最新内核源代码的基础，又急切希望能够进一步理解类 UNIX 操作系统内核工作原理和实际代码实现的爱好者。在阅读时可以参考为本书专门开设的网站 www.oldlinux.org。从中可以下载到很多学习资料和上机实习软件，也可以进行在线讨论。

作　者

目　录

第1章 概 述

本章首先概述 Linux 操作系统诞生的过程和开发、成长的重要环境支柱，解释了选择早期内核版本作为学习对象的原因，并具体说明选择早期内核进行学习的优点，以及如何开始进一步学习。

1.1 Linux 的诞生和发展

Linux 操作系统是 UNIX 操作系统的一种克隆版本。它诞生于 1991 年的 10 月 5 日(这是第一次正式向外公布的时间)。以后借助互联网，在全世界各地计算机爱好者的共同努力下，发展成目前世界上用户最多的一种类 UNIX 操作系统，并且使用人数还在迅猛增加。Linux 操作系统的诞生、发展和成长过程依赖于五个重要支柱：UNIX 操作系统、MINIX 操作系统、GNU 计划、POSIX 标准和互联网。

1.1.1 UNIX、MINIX、GNU 和 POSIX

UNIX 操作系统是美国贝尔实验室的 Ken Thompson 和 Dennis Ritchie 于 1969 年夏在 DEC PDP-7 小型计算机上开发的一个分时操作系统。后经 Dennis Ritchie 于 1972 年用 C 语言进行了改写，使得 UNIX 系统在大专院校得到了推广。

MINIX 操作系统是由 Andrew S. Tanenbaum 在 1987 年编制的，主要用于学生学习操作系统原理。目前主要有两个版本在使用：1.5 版和 2.0 版。最初该操作系统在大学使用是免费的，但其他用户需要付费使用。现在它已经是完全免费的，可以从许多 FTP 上下载。

GNU 计划和自由软件基金会(the Free Software Foundation - FSF)是由 Richard M. Stallman 于 1984 年创办的，旨在开发一个免费的、类似 UNIX 的操作系统——GNU 系统。到 20 世纪 90 年代初，GNU 项目已经开发出许多高质量的免费软件，其中包括有名的 Emacs 编辑系统、BASH shell 程序、GCC 系列编译器程序、GDB 调试程序等。这些软件为 Linux 操作系统的开发创造了一个合适的环境，是 Linux 能够诞生的基础之一。所以目前许多人都将 Linux 操作系统称为“GNU/Linux”操作系统。

POSIX(Portable Operating System Interface for Computing Systems)是由 IEEE 和 ISO/IEC 开发的一簇标准。到 20 世纪 90 年代初，POSIX 标准的制定正处在最后投票敲定的时候。此时 Linux 刚刚起步。这个标准为 Linux 提供了极为重要的信息，使得 Linux 能够在标准的指导下进行开发，做到与绝大多数 UNIX 系统兼容。

1.1.2 Linux 操作系统的诞生和版本的变迁

1981 年 IBM 公司推出著名的 IBM PC 微型计算机。此后十年间，MS-DOS 操作系统一直是微型计算机上的主宰。在这一阶段，硬件价格虽然逐年下降，但软件价格仍然居高不下。当时另一个阵营是 UNIX 世界。为了寻求高利润，UNIX 经销商将价格抬得极高。一度受到贝

尔实验室许可而可以在大学中用于教学的 UNIX 源代码也一直被小心地守卫着不许公开。正在此时，出现了 MINIX 操作系统，并有一本详细描述其工作原理的书可供参考。于是几乎所有计算机爱好者都开始阅读此书。其中也包括 Linux 系统的创始者 Linus Benedict Torvalds。

在 1990 年，20 岁的 Linus 是芬兰赫尔辛基大学计算机系的二年级学生，也是一个自学的电脑黑客。为了学习计算机知识，Linus 用当年圣诞节得到的压岁钱和贷款，购买了一台 386 兼容电脑，并从美国邮购了 MINIX 系统软件。在等待 MINIX 软件期间，Linus 认真研究了 80386 的硬件知识。为了能通过 Modem 拨号连接到学校的主机上，他使用汇编语言并利用 80386 CPU 的多任务特性编制出一个终端仿真程序。为了将自己一台老式电脑上的软件复制到新电脑上，他还为软驱、键盘等一些硬件设备编制出驱动程序。

通过编程实践，并在学习过程中认识到 MINIX 系统的诸多限制，Linus 开始有了编制操作系统的想法。此时 GNU 计划已经开发出许多工具软件，其中最受期盼的 GNU C 编译器已经出现，但 GNU 的操作系统 HURD 仍在开发中。而 MINIX 也有了版权，需要购买才能得到源代码。对于 Linus 来说已经等不及了。从 1991 年 4 月份起，他通过修改终端仿真程序和硬件驱动程序，开始编制起自己的操作系统来。根据 Linus 在 comp.os.minix 新闻组上发布的消息，我们可以知道他逐步从学习 MINIX 系统阶段发展到开发自己的 Linux 系统的过程。

在他回答有关 MINIX 的一个问题时，所说的第一句话就是“请阅读源代码”。他认为答案就在源程序中。这也说明了对于学习系统软件来说，不仅需要懂得系统的基本工作原理，还需要结合源代码，学习系统的实现方法。因为理论毕竟是理论，其中省略了许多细节，而这些细节问题虽然没有太多的理论含量，但它们是一个系统必要的组成部分，就象麻雀身上的羽毛。

第一个与 Linux 系统有关的消息是 Linus 在 1991 年 7 月 3 日在 comp.os.minix 上发布的，其中透露了他正在进行 Linux 系统的开发，并且在最初的 Linux 内核代码中，Linus 就已经为 Linux 与 POSIX 标准的兼容性作好了准备工作。在 0.01 版内核/include/unistd.h 文件中就已经定义了几个 POSIX 标准要求的常数符号。在相关注释中，Linus 写道：“OK，这也许是个玩笑，但我正在着手研究它呢”。到 1991 年的 10 月 5 日，Linus 在 comp.os.minix 新闻组上发布消息，正式向外宣布 Linux 内核系统的诞生(Free MINIX-like kernel sources for 386-AT)。这段消息可以称为 Linux 的诞生宣言，并且一直广为流传

Linux 操作系统从诞生到 1.0 版正式出现，发布的主要版本如表 1-1 所示。

表 1-1　Linux 内核的主要版本

版 本 号	发 布 日 期	说　明
0.01	1991.8	第一个正式向外公布的 Linux 内核版本。多线程文件系统、分段和分页内存管理
0.02	1991.10.5	该版本以及 0.03 版是内部版本，目前已经无法找到特点同上
0.10	1991.10	由 Ted Ts'o 发布的 Linux 内核版本。增加了内存分配库函数
0.11	1991.12.8	基本可以正常运行的内核版本。支持硬盘和软驱驱动
0.12	1992.1.15	主要增加了数学协处理器的软件模拟程序，增加了作业控制、虚拟控制台、文件符号链接和虚拟内存对换功能
0.95(0.13)	1992.3.8	加入虚拟文件系统支持，增加了登录功能。改善了软盘驱动程序和文件系统的性能。改变了硬盘编号方式。支持 CDROM

（续）

版 本 号	发布日期	说 明
0.96	1992.5.12	开始加入网络支持。改善了串行驱动、高速缓冲、内存管理的性能，支持动态链接库，并能运行 X－Window 程序
0.97	1992.8.1	增加了对新的 SCSI 驱动程序的支持
0.98	1992.9.29	改善了对 TCP/IP(0.8.1)网络的支持，纠正了 extfs 的错误
0.99	1992.12.13	重新设计进程对内存的使用分配，每个进程有 4GB 线性空间
1.0	1994.3.14	第一个正式版

将 Linux 系统 0.13 版内核直接改称 0.95 版，Linus 的意思是让大家不要觉得离 1.0 版还很遥远。同时，从 0.95 版开始，对内核的许多改进之处(补丁程序的提供)均以其他人为主了，而 Linus 的主要任务开始变成对内核的维护和决定是否采用某个补丁程序。到现在为止，最新的内核版本是 2003 年 12 月 18 日公布的 2.6.2 版。包括约 15000 个文件，使用 gz 压缩后大小也有 40MB 左右！

1.2 内容综述

本书主要对 Linux 的早期内核 0.11 版进行详细描述和注释。Linux-0.11 版本是在 1991 年 12 月 8 日发布的。在发布时包括以下几个文件：

```
bootimage.Z        - 具有美国键盘代码的压缩启动映像文件；
rootimage.Z        - 以 1200KB 压缩的根文件系统映像文件；
linux-0.11.tar.Z   - 内核源代码文件；
as86.tar.Z         - Linux bruce evans'二进制执行文件，是 16 位的汇编程序和装入程序；
INSTALL-0.11       - 更新过的安装信息文件。
```

目前除了原来的 rootimage.Z 文件，其他四个文件均能找到。不过作者已经利用互联网上的资源为 Linux 0.11 重新制作出了一个完全可以使用的 rootimage-0.11 根文件系统。并重新为其编译出能在 0.11 环境下使用的 gcc1.40 编译器，配置出可用的实验开发环境。目前，这些文件均可以从 oldlinux.org 网站上下载。具体下载位置是：

http://oldlinux.org/Linux.old/images/ 该目录中含有已经制作好的内核映像文件 bootimage 和根文件系统映像文件 rootimage。

http://oldlinux.org/Linux.old/kernels/ 该目录中含有内核源代码程序，包括本书所描述的 Linux 0.11 内核源代码程序。

http://oldlinux.org/Linux.old/bochs/ 该目录中含有已经设置好的运行在计算机仿真系统 bochs 下的 Linux 系统。

http://oldlinux.org/Linux.old/Linux-0.11/ 该目录中含有可以在 Linux0.11 系统中使用的其他一些工具程序和说明文档。

本书分析了 Linux-0.11 内核中所有源代码程序，对每个源程序文件都进行了详细注释，包括对 Makefile 文件和 .h 头文件的注释。分析过程主要是按照计算机启动过程进行的。因此分析的连贯性到初始化结束内核开始调用 shell 程序为止。其余的各个程序均针对其自身进行分析，没有连贯性，因此可以根据自己的需要进行阅读。但在分析时还是提供了一些应用

实例。

在分析过程中，如果第一次遇到较难理解的语句时，将给出相关知识的详细介绍。比如，在阅读代码第一次遇到 C 语言内嵌汇编代码时，将对 GNU C 语言的内嵌汇编语言进行较为详细的介绍；在遇到对中断控制器进行输入/输出操作时，将对 Intel 中断控制器(8259A)芯片给出详细的说明，并列出使用的命令和方法。这样做有助于加深对代码的理解，又能更好地了解所用硬件的使用方法，作者认为这种解读方法要比单独列出一章内容来总体介绍硬件或其他知识效率要高得多。

为了保持结构的完整性，对代码的说明是以内核中源代码的组成结构进行的，基本上是以每个源代码中的目录为一章内容进行介绍。整个 Linux 内核源代码的目录结构见表 1-2。所有目录结构均以 Linux 为当前目录。

表 1-2　Linux 目录

名　称	大　小	最后修改日期
boot/		1991-12-05 22:48:49
fs/		1991-12-08 14:08:27
include/		1991-09-22 19:58:04
init/		1991-12-05 19:59:04
kernel/		1991-12-08 14:08:00
lib/		1991-12-08 14:10:00
mm/		1991-12-08 14:08:21
tools/		1991-12-04 13:11:56
Makefile	2887 bytes	1991-12-06 03:12:46

本书内容可以分为三个部分。第 1 章至第 4 章是描述内核引导启动和 32 位运行方式的准备阶段，学习内核的初学者应该全部进行阅读。第二部分从第 5 章到第 10 章是内核代码的主要部分。其中第 5 章内容可以作为阅读本部分后续章节的索引。第 11 章到第 13 章是第三部分内容，可以作为阅读第二部分代码的参考。

第 2 章概要描述了 Linux 操作系统的体系结构，内核源代码文件放置的组织结构以及每个文件的大致功能。还介绍了 Linux 对物理内存的使用分配方式以及对虚拟线性地址的使用分配，以及如何在 RedHat Linux 9 操作系统上编译本书所讨论的 Linux 内核，对内核代码需要修改的地方。最后开始注释内核程序包中 Linux/目录下所看到的第一个文件，即内核代码的总体 Makefile 文件的内容。该文件是所有内核源程序的编译管理配置文件，供编译管理工具软件 make 使用。

第 3 章将详细注释 boot/目录下的三个汇编程序，其中包括磁盘引导程序 bootsect.s、获取 BIOS 中参数的 setup.s 汇编程序和 32 位运行启动代码程序 head.s。这三个汇编程序完成了内核从块设备上引导加载到内存，对系统配置参数进行探测，完成了进入 32 位保护模式运行之前的所有工作，为内核系统进一步的初始化工作作好了准备。

第 4 章主要介绍 init/目录中内核系统的初始化程序 main.c。它是内核完成所有初始化工作并进入正常运行的关键。在完成了系统所有的初始化工作后，创建了用于 shell 的进程。在介绍该程序时将需要查看其所调用的其他程序，因此对后续章节的阅读可以按照这里调用的顺序进行。由于内存管理程序的函数在内核中被广泛使用，因此该章内容应该最先选读。当你能真正看懂直到 main.c 程序为止的所有程序时，你已经对 Linux 内核有了一定的了解，可以说已经有一

半入门了，但你还需要对文件系统、系统调用、各种驱动程序等进行更深一步的阅读。

第 5 章主要介绍 kernel/目录中的所有程序。其中最重要的部分是进程调度函数 schedule()、sleep_on()函数和有关系统调用的程序。此时你已经对其中的一些重要程序有所了解。

第 6 章对 kernel/dev_blk/目录中的块设备程序进行了注释说明。该章主要含有硬盘、软盘等块设备的驱动程序，主要与文件系统和高速缓冲区打交道。因此，在阅读这章内容时需首先浏览一下文件系统的章节。

第 7 章对 kernel/dev_chr/目录中的字符设备驱动程序进行注释说明。这一章中主要涉及串行线路驱动程序，键盘驱动程序和显示器驱动程序，因此含有较多与硬件有关的内容，在阅读时需要参考一下相关书籍。

第 8 章介绍 kernel/math/目录中的数学协处理器的仿真程序。由于本书所注释的内核版本，还没有真正开始支持协处理器，因此本章的内容较少，也比较简单，只需有一般性的了解即可。

第 9 章介绍内核源代码 fs/目录中的文件系统程序，在看这章内容时建议你能够暂停一下，先去阅读 Andrew S. Tanenbaum 的《操作系统：设计与实现》一书中有关 MINIX 文件系统的章节，因为最初的 Linux 系统只支持 MINIX 文件系统，Linux 0.11 版也是如此。

第 10 章解说 mm/目录中的内存管理程序。要透彻地理解这方面的内容，需要对 Intel 80x86 微处理器的保护模式运行方式有足够的理解，因此本章在适当的地方包含较为完整的有关 80x86 保护模式运行方式的说明，这些知识基本上都可以参考 Intel 80386 程序员编程手册(Intel 80386 Programmer's Reference Manual)。但在此章中，以源代码中的运用实例为对象进行解说，可以更好地理解它的工作原理。

现有的 Linux 内核分析书籍都缺乏对内核头文件的描述，因此对于一个初学者来讲，在阅读内核程序时会遇到许多障碍。本书的第 11 章对 include/目录中的所有头文件进行了详细说明，基本上对每一个定义、每一个常量或数据结构都进行了详细注释。虽然该章内容主要是为阅读其他章节中的程序作参考使用的，但是若想彻底理解内核的运行机制，仍然需要了解这些头文件中的许多细节。

第 12 章介绍了 Linux 0.11 版内核源代码 lib/目录中的所有文件。这些库函数文件主要向编译系统等系统程序提供接口函数，阅读这些文件对以后理解系统软件会有较大的帮助。由于这个版本较低，所以这方面的内容并不是很多，可以很快地看完。这也是我们为什么选择 0.11 版的原因之一。

第 13 章介绍 tools/目录下的 build.c 程序。这个程序并不包括在编译生成的内核映像(image)文件中，它仅用于将内核中的磁盘引导程序块与其他主要内核模块连接成一个完整的内核映像(kernel image)文件。

1.3 本章小结

本章首先阐述了 Linux 诞生和发展不可缺少的五个支柱：UNIX 最初的开放源代码版本为 Linux 提供了基本原理和算法；Richard Stallman 的 GNU 计划为 Linux 系统提供了丰富且免费的各种实用工具；POSIX 标准的出现为 Linux 提供了实现与标准兼容系统的参考指南；Tanenbaum 的 MINIX 操作系统为 Linux 的诞生起到了不可忽略的参考作用；互联网是 Linux 成长和壮大的必要环境。

1.4 习题

1. 利用互联网上的搜索引擎，查找有关 UNIX、MINIX、GNU 的主要相关站点和详细信息。并使用 www.goolge.com 网站上的 Linux 新闻组，了解 Linux 诞生和发展的过程。

2. 复习有关 Intel 80x86 体系结构的硬件及其编程知识，尤其是关于 Intel 80x86 运行在保护虚拟地址模式(简称保护模式)下的工作原理。

3. Linus 在最初开发 Linux 操作系统内核时，主要算法参考了 M. J. Bach 著的《UNIX 操作系统设计》一书，Linux 内核源代码中很多重要函数的名称都取自该书。在继续学习本书之前，请先复习一下该书的基本内容。

第 2 章　Linux 内核体系结构

本章首先基于 Linux 0.11 版的内核源代码，简要描述内核的基本体系结构、主要构成模块，概要说明了内核源代码目录中组织形式以及子目录中各个代码文件的主要功能、基本调用的层次关系。随后描述了构建 Linux 0.11 内核编译实验环境的方法。接下来切入正题，从内核源文件 linux/目录下的第一个文件 Makefile 开始，对每一行代码进行详细说明。

2.1　Linux 内核模式和体系结构

一个完整可用的操作系统主要由 4 部分组成：硬件、操作系统内核、操作系统服务和用户应用程序，如图 2-1 所示。用户应用程序是指那些字处理程序、互联网浏览器程序或用户自行编制的各种应用程序；操作系统服务程序是指向用户提供的服务，被看作是操作系统部分功能的程序。在 Linux 操作系统上，这些程序包括 X 窗口系统、shell 命令解释系统以及内核编程接口等系统程序；操作系统内核程序是本书所感兴趣的部分，它主要用于对硬件资源的抽象和访问调度。

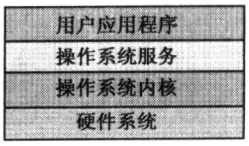

图 2-1　操作系统组成部分

目前，操作系统内核的结构模式主要可分为整体式的单内核模式和层次式的微内核模式。而本书所注释的 Linux 0.11 内核，则采用了单内核模式。单内核模式的主要优点是内核代码结构紧凑、执行速度快，不足之处主要是层次结构性不强。

在单内核模式系统中，操作系统提供服务的流程为：应用主程序使用指定的参数执行系统调用指令(int x80)，使 CPU 从用户态(User Mode)切换到核心态(Kernel Mode)，然后系统根据参数值调用特定的系统调用服务程序，而这些服务程序则根据需要调用底层的支持函数以完成特定的功能。在完成了应用程序要求的服务后，操作系统又从核心态切换回用户态，回到应用程序中继续执行后续指令。因此，单内核模式的内核也可粗略地分为三层：调用服务的主程序层、执行系统调用的服务层和支持系统调用的底层函数，如图 2-2 所示。

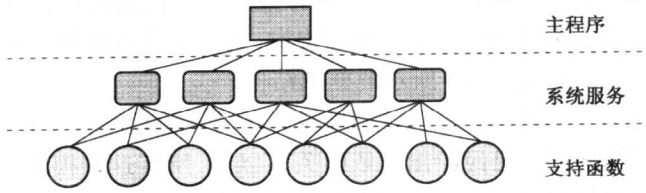

图 2-2　单内核模式的简单结构模型

Linux 内核主要由 5 个模块构成，它们分别是：进程调度模块、内存管理模块、文件系统模块、进程间通信模块和网络接口模块。

进程调度模块用来负责控制进程对 CPU 资源的使用。所采取的调度策略使各进程能够

公平合理地访问 CPU，同时保证内核能及时地执行硬件操作。内存管理模块用于确保所有进程能够安全地共享机器主内存区，同时，内存管理模块还支持虚拟内存管理方式，使 Linux 的进程可以使用比实际内存空间更多的内存容量。并可以利用文件系统把暂时不用的内存数据块交换到外部存储设备上去，当需要时再交换回来。文件系统模块用于支持对外部设备的驱动和存储。虚拟文件系统模块通过向所有的外部存储设备提供一个通用的文件接口，隐藏了各种硬件设备的不同细节。从而提供并支持与其他操作系统兼容的多种文件系统格式。进程间通信模块子系统用于支持多种进程间的信息交换方式。网络接口模块提供对多种网络通信标准的访问并支持许多网络硬件。

这几个模块之间的依赖关系见图 2-3。其中的连线代表它们之间的依赖关系，虚线和虚框部分表示 Linux 0.11 中还未实现的部分(从 Linux 0.95 版才开始逐步实现虚拟文件系统，而网络接口的支持到 0.96 版才有)。

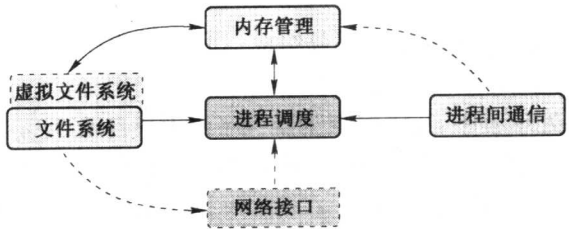

图 2-3　Linux 内核系统模块结构及相互依赖关系

由图可以看出，所有的模块都与进程调度模块存在依赖关系。因为它们都需要依靠进程调度程序来挂起(暂停)或重新运行它们的进程。通常，一个模块会在等待硬件操作期间被挂起，而在操作完成后才可继续运行。例如，当一个进程试图将一数据块写到软盘上去时，软盘驱动程序就可能在启动软盘加速期间将该进程置为挂起等待状态，而在软盘进入正常转速后再使得该进程能继续运行。另外 3 个模块也是由于类似的原因而与进程调度模块存在依赖关系。

其他几个依赖关系有些不太明显，但同样很重要。进程调度子系统需要使用内存管理器来调整一特定进程所使用的物理内存空间。进程间通信子系统则需要依靠内存管理器来支持共享内存通信机制。这种通信机制允许两个进程访问内存的同一个区域以进行进程间信息的交换。虚拟文件系统也会使用网络接口来支持网络文件系统(NFS)，同样也能使用内存管理子系统来提供内存虚拟盘(ramdisk)设备。而内存管理子系统也会使用文件系统来支持内存数据块的交换操作。若从单内核模式结构模型出发，我们还可以根据 Linux 0.11 内核源代码的结构将内核主要模块绘制成图 2-4 所示的框图结构。

其中内核级中的几个方框，除了硬件控制方框以外，其他粗线方框分别对应内核源代码的目录组织结构。除了这些图中已经给出的依赖关系以外，所有这些模块还会依赖于内核中的通用资源。这些资源包括内核所有子系统都会调用的内存分配和收回函数、打印警告或出错信息函数以及一些系统调试函数。

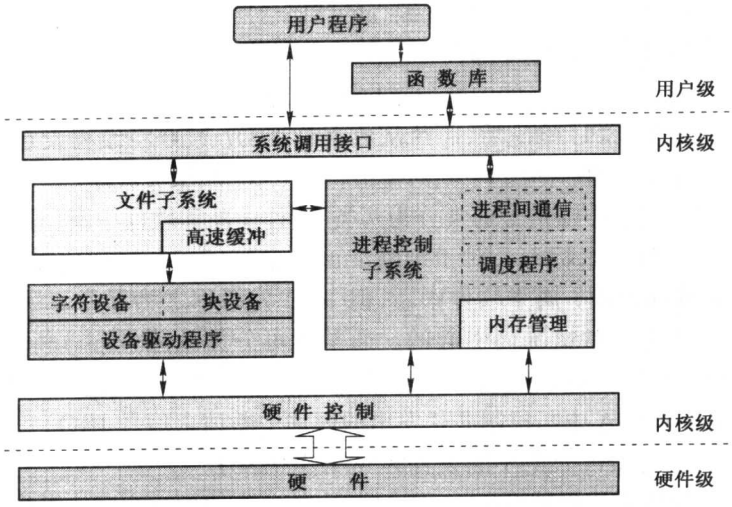

图 2-4　内核结构框图

2.2　Linux 中断机制

在使用 80x86 组成的 PC 机中，采用了两片 8259A 可编程中断控制芯片。每片可以管理 8 个中断源。通过多片的级联方式，能构成最多管理 64 个中断向量的系统。在 PC/AT 系列兼容机中，使用了两片 8259A 芯片，共可管理 15 级中断向量。其级连示意图见图 2-5。其中从芯片的 INT 引脚连接到主芯片的 IR2 引脚上。主 8259A 芯片的端口基地址是 0x20，从芯片是 0xA0。

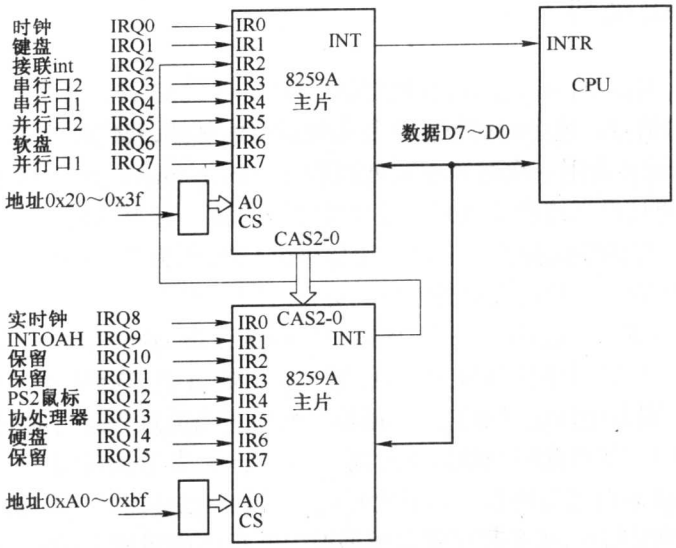

图 2-5　PC/AT 微机级连式 8259 控制系统

在总线控制器控制下，8259A 芯片可以处于编程状态和操作状态。编程状态是 CPU 使用

IN 或 OUT 指令对 8259A 芯片进行初始化编程的状态。一旦完成了初始化编程，芯片即进入操作状态，此时芯片即可随时响应外部设备提出的中断请求(IRQ0～IRQ15)。通过中断判优选择，芯片将选中当前最高优先级的中断请求作为中断服务对象，并通过 CPU 引脚 INT 通知 CPU 外中断请求的到来，CPU 响应后，芯片从数据总线 D7～D0 将编程设定的当前服务对象的中断号送出，CPU 由此获取对应的中断向量值，并执行中断服务程序。

对于 Linux 内核来说，中断信号通常分为两类：硬件中断和软件中断(异常)。每个中断是由 0～255 之间的一个数字来标识。对于中断 int0～int31(0x00～0x1f)，每个中断的功能由 Intel 公司固定设定或保留用，属于软件中断，但 Intel 公司称之为异常。因为这些中断是在 CPU 执行指令时探测到异常情况而引起的。通常还可分为故障(fault)和陷阱(traps)两类。中断 int32～int255 (0x20～0xff)可以由用户自己设定。在 Linux 系统中，则将 int32～int47 (0x20～0x2f)对应于 8259A 中断控制芯片发出的硬件中断请求信号 IRQ0～IRQ15，并把程序编程发出的系统调用(system_call)中断设置为 int128(0x80)。

在 Linux 0.11 内核源代码的 head.s 程序中，内核首先使用一个哑中断向量(中断描述符)对中断描述符表(Interrupt Descriptor Table - IDT)中所有 256 个描述符进行了默认设置(boot/head.s,78)。这个哑中断向量指向一个默认的“无中断”处理过程(boot/head.s,150)。当发生了一个中断而又没有重新设置过该中断向量时就会显示信息“未知中断(Unknown interrupt)”。因此，对于系统需要使用的一些中断，内核必须在其继续初始化的处理过程中(init/main.c)重新设置这些中断的中断描述符项，让它们指向对应的实际处理过程。通常，硬件异常中断处理过程(int0～int 31)都在 traps.c 的初始化函数中进行了重新设置(kernel/traps.c,181)，而系统调用中断 int128 则在调度程序初始化函数中进行了重新设置(kernel/sched.c,385)。

2.3 Linux 系统定时

在 Linux 0.11 内核中，PC 机的可编程定时芯片 Intel 8253 被设置成每隔 10ms 就发出一个时钟中断(IRQ0)信号。这个时间节拍就是系统运行的脉搏，我们称之为 1 个系统滴答。因此每经过 1 个滴答就会调用一次时钟中断处理程序(timer_interrupt)。该处理程序主要用来通过 jiffies 变量来累计自系统启动以来经过的时钟滴答数。每当发生一次时钟中断该值就增 1。然后从被中断程序的段选择符中取得当前特权级 CPL 作为参数调用 do_timer()函数。

do_timer()函数则根据特权级对当前进程运行时间作累计。如果 CPL = 0，则表示进程是运行在内核态时被中断，因此把进程的内核运行时间统计值 stime 增 1，否则把进程用户态运行时间统计值增 1。如果程序添加过定时器，则对定时器链表进行处理。若某个定时器时间到(递减后等于 0)，则调用该定时器的处理函数。然后对当前进程运行时间进行处理，把当前进程运行时间片减 1。如果此时当前进程时间片还大于 0，表示其时间片还没有用完，于是就退出 do_timer()继续运行当前进程。如果此时进程时间片已经递减为 0，表示该进程已经用完了此次使用 CPU 的时间片，于是程序就会根据被中断程序的级别来确定进一步处理的方法。若被中断的当前进程是工作在用户态的(特权级别大于 0)，则 do_timer()就会调用调度程序 schedule()切换到其他进程去运行。如果被中断的当前进程工作在内核态，即在内核程序中运行时被中断，则 do_timer()会立刻退出。因此这样的处理方式决定了 Linux 系统在内核态运