Learning Linux Kernel by Practice

李善平　陈文智　等　编著

边干边学
——LINUX内核指导

浙江大學出版社

边 干 边 学

——LINUX 内核指导

李善平　陈文智 等 编著

浙江大學出版社

图书在版编目（CIP）数据

边干边学：LINUX 内核指导 / 李善平，陈文智等编著.
杭州：浙江大学出版社，2002.8
ISBN 7-308-03073-3

Ⅰ.边... Ⅱ.①李...②陈... Ⅲ.Linux 操作系统
—基本知识 Ⅳ.TP316.89

中国版本图书馆 CIP 数据核字（2002）第 051037 号

责任编辑：杜希武
封面设计：俞亚彤
出版发行：浙江大学出版社
（杭州浙大路 38 号 邮政编码 310027）
（E-mail:zupress@mail.hz.zj.cn）
（网址：http://www.zjupress.com）
排　　版：浙江大学出版社电脑排版中心
印　　刷：浙江印刷集团公司
开　　本：889mm×1194mm 1/16
印　　张：30
字　　数：846 千
版、印次：2002 年 8 月第 1 版 2003 年 3 月第 2 次印刷
印　　数：3001—6000
书　　号：ISBN 7-308-03073-3/TP・234
定　　价：45.00 元

前　言

学计算机专业知识必须动手实践，这点应该不会有异议吧。

那么学操作系统怎么实践呢？尤其是我们近年来承担了浙江大学计算机学院操作系统类课程的教学，更需认真回答这个问题了。

首先，选择实验对象和环境。这一点已经找到答案了，那就是 Linux 操作系统内核。我们在 Linux 内核分析方面作了大量的研究，也编写了几本教材。

其次，搜集指导材料。我们自己在学习 Linux 过程中，耗费了许多精力和时间搜集资料，又耗费了许多精力和时间分析、整理这些资料。如果您不是天才，同样也要耗费精力和时间再去做这些。但现在您可以不必这样了，因为我们可以共享。

对了，这就是我们费劲写这本书的起因。我们希望奉上这些心得，帮助您在学习 Linux 内核时事半功倍。

最后，学了不用仍然白搭。运用 Linux 操作系统，我们正在嵌入式系统和应用服务器方面进行研究和开发，有机会也将这方面的体会一并奉上。

当然，阅读后敬请提出宝贵意见。

全书由李善平、陈文智共同负责编写。其中，李善平编写第 1，2，5，7，9，10 章，陈文智编写第 3，4，6，8，11 章。需要特别强调的是，浙江大学计算机学院的部分研究生为此书的出版付出了大量的时间和精力，江晖的小组参加了第 1，2 章的编写，纪舟的小组参加了第 3 章的编写，马轶平的小组参加了第 4 章的编写，尹康凯的小组参加了第 5 章的编写，卓亚芬的小组参加了第 6 章的编写，金海荣的小组参加了第 7 章的编写，江峰的小组参加了第 8 章的编写，单宏伟、余峰的小组参加了第 9 章的编写，袁洋的小组参加了第 10 章的编写，刘敏、李岩和王总辉参加了第 11 章的编写。

P1517305

目　录

试读结束，需要全本PDF请购买 www.ertongbook.com

第1章

了解Linux内核

Linux是当今流行的操作系统之一。由于其源码的开放性，现代操作系统设计的思想和技术能够不断运用于它的新版本中。因此，读懂并修改Linux内核源代码无疑是学习操作系统设计技术的有效方法。本章介绍Linux内核的概况以及查看系统运行状况的方法。首先介绍Linux内核的特点、源码结构和重新编译内核的方法。第二节讲述如何通过Linux系统所提供的/proc虚拟文件系统了解操作系统运行状况的方法。在第三节我们将编程实现一个系统状况观察工具。最后一节对Linux编程环境中的常用工具作一简单介绍。

1.1 Linux内核

Linux是类UNIX操作系统的一个分支，它最初是由Linus Torvalds于1991年为基于Intel 80386的IBM兼容机开发的操作系统。在加入自由软件组织GNU后，经过Internet上全体开发者的共同努力已成为能够支持各种体系结构（包括Alpha、SPARC、Motorola、MC680x0、PowerPC、IBM System/390等）的具有很大影响的操作系统。Linux最大的优势在于它不是商业操作系统，其源代码受GNU通用公共许可证（GPL）的保护，是完全开放的，任何人都能下载下来用于研究和开发。

通过学习Linux，你可以体会到一个现代的操作系统是如何设计实现的。我们的目的就是指引你进入这个神秘的境地去探索操作系统的奥秘。

1.1.1 什么是Linux内核

Linux其实只是一个内核的标识，不同于我们平时所说的RedHat Linux、Debian GNU/Linux等发行版本，这些发行版本除了内核外还包括了不同的外部应用程序以方便用户使用和管理操作系统。内核（kernel）是操作系统的内部核心程序，它向外部提供了对计算机设备的核心管理调用。一般来讲，操作系统上运行的代码可以分成两部分：内核所在的地址空间称作内核空间；而在内核以外，剩下的程序统称为外部管理程序，它们大部分是对外围设备的管理和界面操作。外部管理程序与用户进程所占据的地址空间称为外部空间或者用户空间。通常，一个程序会跨越两个空间。当执行到内核空间的

一段代码时，我们称程序处于内核态，而当程序执行到外部空间代码时，我们称程序处于用户态。内核负责对计算机硬件的管理和抽象，并合理分配这些资源给各个执行程序共享使用。Linux 内核主要包括以下功能：

- 资源抽象　用软件接口抽象不同硬件资源简化对其的操作，屏蔽底层硬件的不同接口。例如，一个设备驱动程序就是对物理设备进行输入/输出操作的软件抽象。一旦实现设备的驱动程序，其他软件就可以通过它读写设备而不必去了解怎样对设备控制寄存器写命令和数据传输等细节问题。抽象还能针对一些没有特殊下层硬件的资源，比如消息和信号量资源。
- 资源分配　经过编译连接后的目标程序只有放入内存（RAM）中才能被执行，Linux 和其他操作系统一样将目标程序执行时的 CPU 操作抽象成进程。进程可以被定义为一个程序执行时的实例或执行上下文。Linux 是多任务（multiprogrammed）操作系统，许多任务能同时在内存中，一个任务执行一段时间片后，操作系统能把 CPU 等资源分配给另一个任务执行。抽象的进程使操作系统能够控制管理每个程序的执行实例，如哪个进程被立即执行而另外的处于等待之中。资源管理就是将抽象出来的各种资源分配给各个进程并负责取回这些系统资源。
- 资源共享　每个进程都能向内核申请得到资源，然后使用并释放它们，这样必然存在竞争现象。一般当一个进程得到系统资源后，它需要独占这些资源。内核根据不同的资源类型使用不同机制保证资源被进程所独占。有些系统资源允许两个以上的进程同时共享访问，如多个进程可以同时打开同一文件。这时内核必须确保各个进程互斥访问共享资源，并负责在所有进程释放资源时的回收工作。

Linux 与大部分 UNIX 内核一样是单内核体系结构(monolithic kernel)的，即它是由几个逻辑功能上不同的部分组合而成的大程序。与之对应的是微内核体系结构，这种内核只包括同步原语、简单的进程调度以及进程间通信机制等功能，其他像内存管理、设备驱动和系统调用功能是由在微内核之上的一些系统进程实现的。一般而言，微内核相对于单内核来说要慢，因为在操作系统各层间调用时的消息传递必定会有一定的消耗。但微内核有模块化、易于移植到其他体系结构以及占用内存比单一内核少等优点。Linux 使用“模块”（module）来有效弥补单一内核的缺点，同时避免了引入微内核而带来的性能损失。模块是在运行时能够被动态链接到内核的目标文件，它们一般用于诸如文件系统的实现、设备驱动程序等属于内核上层的功能代码。与微内核中的外层部分不同，模块不是一个独立的进程而是与其他静态链接到内核的功能一样在内核模式下执行。关于模块的原理和应用，第 4 章有非常详尽的讨论。

Linux 并不支持完全意义上的用户态线程。线程是同时执行的共享资源的程序段。线程之间可以共享地址空间、物理内存页面，甚至打开的设备和文件。这样，在线程间切换要比在进程间切换的开销少，大量使用线程可以使系统的效率得到提高。所以，线程在现代操作系统中得到了广泛的应用。但 Linux 中线程的使用却很少见到，只是在内核态中定期执行某些函数时才会用到线程的概念。在用户态中，Linux 通过另一种方法解释并实现 LWP（light weight thread）的机制。Linux 中认为线程就是共享上下文（context）的进程，并可以通过非标准的系统调用 clone()来处理。

Linux 的内核为非抢占式的（non-preemptive）。这就是说 Linux 在特权级执行时不会任意改变执行流程。这样，许多 Linux 内核代码中可以对某些重要的数据结构进行修改而不加任何保护措施，因为内核不必担心被其他程序所抢占。

另外，Linux 内核还支持对称多处理器（symmetric multiprocessing，简称 SMP）结构。这样，任意处理器可以执行任意程序。但是，Linux 中并没有针对 SMP 结构进行优化，就是说许多内核代码（如文件系统处理和网络）能够同时在不同处理器上运行，但必须保证它们的顺序执行。本书所讨论的内容都是基于单处理器结构。

Linux 符合 IEEE 的 POSIX(Portable Operating System Interface based on Unix)标准，为用户程序提供了规范的应用编程接口(Application Programming Interface,简称 API)。用户程序大部分时间执行在用户模式下，它们不能直接访问内核态的数据也不能直接对硬件进行操作（某些操作系统如 DOS 可以允许用户对硬件直接访问）。它们只能通过内核提供的标准编程接口即系统调用请求内核服务，并切换到内核态执行。当内核完成用户请求，再将用户程序返回到用户态。图 1-1 反映了 Linux 内核与用户进程及硬件之间的关系。

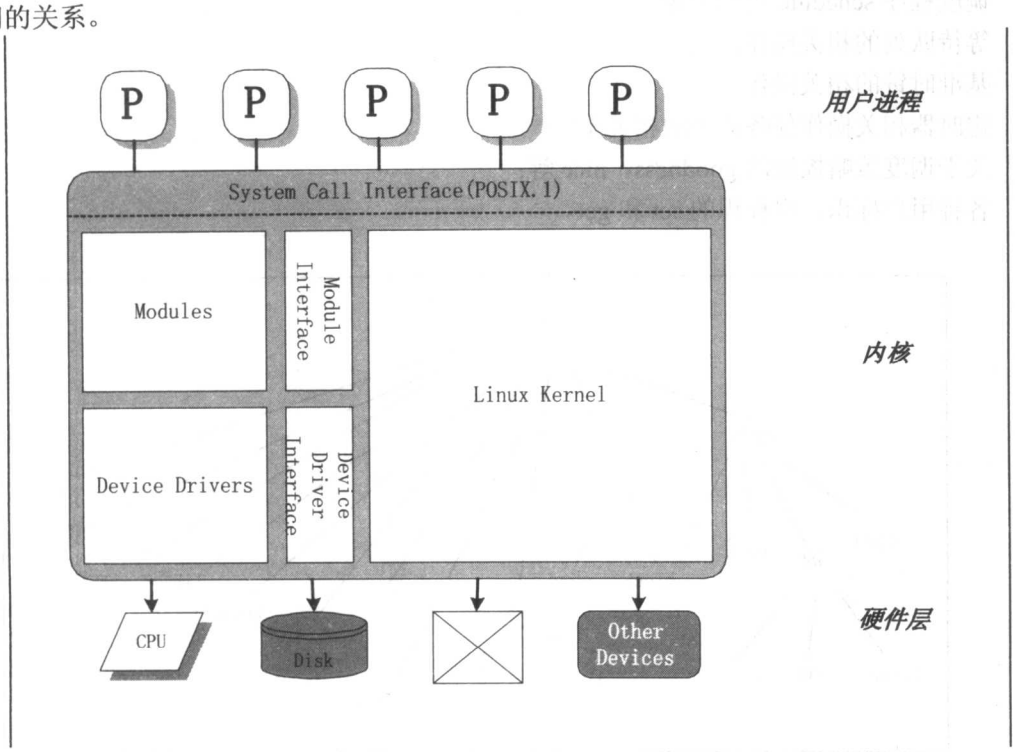

图 1-1　Linux 内核的结构

1.1.2　内核版本及目录结构

Linux 可以通过版本号简单地区分内核是稳定版本还是开发测试版本。每个版本号都由用点号分隔的三个数组成，前两个数表示版本，最后一个数表示发行号。第二个数是偶数则表示是稳定的版本，否则是开发测试版本。如我们在本书中使用的内核源码版本是稳定版本 2.4.18,而版本 2.3.99 和 2.5.6 都是开发测试版内核。同一稳定版本的新发行号只是解决了一些用户报告的 bug，数据结构和算法实现一般都没有太大变化。而开发测试版本则可能在很多方面有改变，因为内核开发者可以自由地使用不同解决方法改变内核。

Linux 的源代码被组织成树形结构，以 linux 为根，其目录结构如图 1-2 所示。一般我们都将它安装在/usr/src/linux 目录下，当然你也可以将它放在你想放的任意目录中（不特别说明，本书中所有代码都相对于/usr/src/linux 目录）。图中显示了 linux 目录下主要的目录和文件。内核的核心函数源码主要在 kernel 和 arch/<体系结构类型>/kernel 两个目录下，arch/<体系结构类型>目录下是与体系结构相关的代码，如我们一般使用的 Intel 80x86 体系结构，则<体系结构类型>就是 i386。下面是对主要目录和文件的一些概述。

1.1.2.1 kernel 目录

在 kernel 目录下文件实现了大多数 Linux 系统的核心函数，其中最重要、最主要的文件当属 sched.c。sched.c 文件定义的函数有：

- 调度程序 schedule 及相关操作；
- 等待队列的相关操作；
- 基准时钟的相关操作；
- 定时器相关操作任务队列的相关操作；
- 关于调度策略控制的 goodness，nice 等；
- 各种用户标识、组标识的 set 和 get。

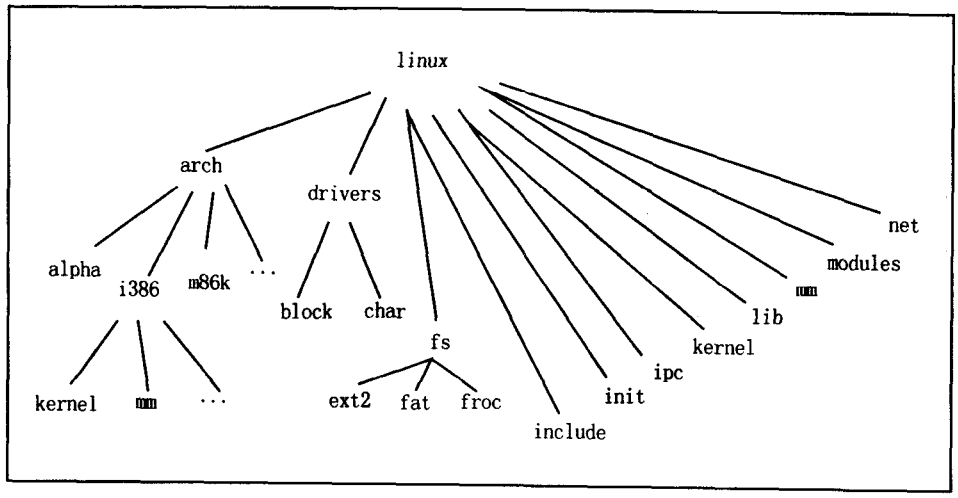

图 1-2 Linux 内核源代码目录结构

关于进程控制的文件也位于此目录下。fork.c 文件定义创建、克隆子进程的函数 do_fork()。exit.c 文件定义结束自身进程的 do_exit()等操作。signal.c 文件中的函数都是关于信号控制的，如发送信号（signal）操作 send_sig。

ksyms.c 文件定义了 Linux 内核的输出符号表，使用模块功能时经常需要对这些符号表操作以实现模块和内核的动态链接。

此外，该目录下还包含一些重要的文件，如：

- time.c，提供用户程序与系统之间关于时间的操作界面；
- resource.c，关于 I/O 端口资源（port）的管理；
- dma.c，关于 DMA 通道的管理；
- softirq.c，关于 bottom half 队列、softirq 等的操作；
- itimer.c，关于 itimer 定时器读写的系统调用；
- printk.c，展示系统工作参数的操作，如 printk。

1.1.2.2 mm 目录

Linux 中独立于 CPU 体系结构特征的内存管理文件几乎都集中在 mm 目录下。如页式存储管理，内存的分配和释放等等。

- swap.c 文件并不实现任何交换算法，它仅仅处理命令行选项“swap=”和“buff=”。
- swapfile.c 文件是管理交换文件和交换设备的源程序。它包含 swapon、swapoff 系统调用的执行程序，以及从交换空间申请空闲页面的操作 get_swap_page。
- page_io.c 文件则实现了与交换空间低层的数据传输。
- swap_state.c 文件维护 swap cache，它包含存储管理系统中最复杂、最难懂的操作和结构。
- vmscan.c 文件定义后台交换进程 kswapd 的代码，以及内存空间扫描函数，实现 Linux 的页面换出策略，如 try_to_free_pages。

所有的存储分配策略都在 mm 目录里实现。例如，slab.c 实现核心内存块（以数据结构 free_area 表为入口）的操作 kmalloc()、kfree()等。vmalloc.c 里包含__vmalloc()、vfree()等函数。物理页面的申请函数源程序集中在 page_alloc.c，如页面申请函数__get_free_pages()。

内存管理中低层核心函数大多安排在 memory.c 文件，如缺页中断响应函数 do_no_page()，实现 Copy on Write（COW）特性的 do_wp_page()，以及众多页表管理函数。

虚拟空间映射（mapping）操作 do_mmap_pgoff()和 do_munmap()，以及系统调用 brk 的响应函数，都涉及进程虚拟空间地址的调整，相关源代码在 mmap.c 文件中。mremap 的操作代码则在 mremap.c 文件中。

此外，mlock.c 文件实现四个关于内存 vma 段加锁操作的系统调用 mlock、munlock、mlockall、munlockall。mprotect.c 实现系统调用 mprotect。

1.1.2.3　fs 目录

文件操作是所有 UNIX 系统都提供的基本功能。fs 目录源程序涵盖各种类型的文件系统，各种类型的文件操作。

本目录下的每个子目录则是分门别类地描述某个特定的文件系统。

直接隶属本目录的文件分别是：

- exec.c。实现 execve 系统调用。其余五种关于装入执行程序的函数都由 C 语言库文件实现。execve 支持脚本（script）文件和多种格式的可执行文件。
- devices.c。负责设备的注册和注销，定义缺省的打开设备操作和释放设备操作。
- block_dev.c。包含缺省的读、写设备操作。
- super.c。定义超级块的读操作，以及文件系统的安装、卸载操作。
- inode.c。VFS inode 的读写操作，以及维护 inode cache 的程序。
- dcache.c。维护 dcache 的文件。
- namei.c。访问权限检查。根据路径检索 dentry 的相关操作，如 open_namei、path_walk。
- buffer.c。实现 buffer cache 的文件。
- open.c。文件的打开、关闭操作。系统调用 chown、chmod、fchown、fchmod、chroot、chdir、fchdir 等也由该文件实现。
- read_write.c。系统调用 read、write、lseek、llseek 的源程序。
- readdir.c。读目录项的系统调用 readdir 和 getdents 由此文件实现。
- select.c。集中存放 select 操作的几乎全部源代码。
- pipe.c 和 fifo.c。实现管道（pipe）和命名管道（fifo）。除了 fifo_open()和 fifo_init()函数在 fifo.c 文件外，其余函数均存放于 pipe.c 文件。
- ioctl.c。实现系统调用 ioctl。
- fcntl.c。实现关于 fcntl 操作命令的源代码。

试读结束，需要全本PDF请购买 www.ertongbook.com

- dquot.c。支持磁盘配额机制（quota）。

1.1.2.4 arch 目录

与 CPU 类型相关的子目录和文件均集中安排在 arch 目录下。这里又有子目录 alpha、i386、m68k、mips、ppc、sparc，每个子目录对应一种 CPU，例如“arch/i386”就是关于 INTEL CPU 的子目录。

1.1.2.5 include 目录

容纳 Linux 源程序的大多数是头文件（header file）。其中，与平台无关的头文件在 include/linux 子目录下，与 Intel CPU 相关的头文件在 include/asm-i386 子目录下。另外，还有关于 SCSI 设备的头文件目录 include/scsi 和关于网络设备的头文件目录 include/net。

除上述目录外，Linux 源程序清单中还有 net、ipc、drivers、init 等目录，其内容就不再一一介绍了。

1.1.3 重新编译内核

重新编译内核是一件比你想像的还要简单的事情，它甚至不需要你对内核有任何了解，只要你具备一些基本的 Linux 操作系统的知识就可以进行。

1.1.3.1 查找并且下载一份内核源代码

我们知道 Linux 的内核代码受 GNU 通用公共许可证（GPL）保护，是完全开放的，现在很多 Linux 的网站都提供内核代码的下载。例如，Linux 的官方网站 www.kernel.org ，在这里你可以找到所有的内核版本。进入 www.kernel.org/pub/linux/kernel/ ，你会看到很多目录。我们在 www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.18.tar.gz 或者在 www.kernel.org /pub/linux/kernel/v2.4/linux-2.4.18.tar.bz2 下载一份 2.4.18 的内核。然后用下面的命令把它放到某个目录（这里假设是 home 目录），并进行解压：

先将压缩包移到你所要放的目录下：

```
# mv linux-2.4.18.tar.gz /home/<dst directory>/
# tar zxvf linux-2.4.18.tar.gz
```

或者，如果下载的是 linux-2.4.18.tar.bz2，那么解压的命令是：

```
# bzip2 -dc linux-2.4.18.tar.bz2 | tar xvf -
```

解压出来的是一个 linux 目录，里面就是 2.4.18 的内核源代码。

1.1.3.2 配置内核

在进行这项工作之前，不妨先看一看刚才我们解压出来的内核源代码目录下的 README 文件。在这份文件中，对怎样进行内核的解压、配置、安装都进行了详细的讲解。

在编译内核前，一般来说都需要对内核进行相应的配置。配置是精确控制新内核功能的机会。配置过程也控制哪些需编译到内核的二进制映象中(在启动时被载入)，哪些是需要时才装入的内核模块

（module）。

首先，重新建立一个核心需要将源代码树置于一种完整和一致的状态。因此，我们推荐执行命令 make mrproper。它将清除目录下所有配置文件和先前生成核心时产生的中间文件：

```
#cd linux
#make mrproper
```

之后，我们就可以对内核进行配置了。有几种方法可以做这件事情：

```
#make config
#make menuconfig
#make xconfig
#make oldconfig
```

下面分别对它们进行简单的解释：

make config 是基于文本的传统配置界面，该配置方式已逐渐被更为简单的 make menuconfig 命令所取代，建议不要用这种方式。

make menuconfig 是基于文本的选单式配置界面，是最为灵活的内核配置工具。我们一般使用这一配置命令。

make xconfig 是基于图形窗口模式的配置界面，也是常用的配置命令，但它需要 X Window 图形环境的支持。

make oldconfig 用于在原来内核配置的基础上作些小的修改。

好了，现在你可以选择你最喜欢的一种方式进行内核配置。

进行配置时，大部分选项可以使用其缺省值，只有小部分需要根据用户不同的需要进行选择。例如，系统如果配有 SCSI 卡等，需要在网络配置中选择 SCSI 卡的支持。

对每一个配置选项，用户有三种选择，它们分别代表的含义如下：

“Y”——将该功能编译进内核；

“N”——不将该功能编译进内核；

“M”——将该功能编译成可以在需要时动态插入到内核中的模块。

将与核心其他部分关系较远且不经常使用的部分功能代码编译成为可加载模块，有利于缩减内核的，减小内核消耗的内存，简化该功能相应的环境改变时对内核的影响。许多功能都可以这样处理，例如，上面提到的对 SCSI 卡的支持、对 ext2 等文件系统的支持等等。

1.1.3.3　编译内核和模块

配置完内核，接下来需要运行下列两条命令清除源代码树，产生相关文件：

```
#make dep
#make clean
```

其中 make dep 对内核源代码文件的依赖性和完整性进行检验，确保关键文件在正确的位置。make clean 清除目标文件，确保所有有关文件都处于最新版本状态。

下面的命令将编译产生压缩形式的内核文件：

```
#make zImage
```

在需要内核支持较多的外设和功能时，内核可能变得很大，此时可以用以下命令编译大内核产生压缩率更高的内核文件：

```
#make bzImage
```

编译内核需要较长的时间（通常要几十分钟），具体与机器的硬件条件及内核的配置等因素有关。完成后产生的内核文件的位置在/usr/src/linux/arch/i386/boot 目录下，当然这里假设用户的 CPU 是 Intel X86 型的，并且你将内核源代码放在/usr/src/linux 目录下。

如果选择了可加载模块，编译完内核后，要对选择的模块进行编译：

```
#make modules
```

或用下面的命令编译模块并安装到标准的模块目录中：

```
#make modules_install
```

模块在系统中的标准目录位于/lib/modules/x.y.z，后面的 x.y.z 是版本号，为安全起见，在运行#make modules_install 之前最好对/lib/modules 进行备份。模块通常是带有扩展名.o 的文件，使用命令#lsmod 可以了解当前内核中已经装载了哪些模块。如果你想把编译的内核移到另一个 Linux 系统中，这个目录也必须被移动并放到目标系统的相同位置。

1.1.3.4 配置启动文件

通常，Linux 在系统引导后从/boot 目录下读取内核映象到内存中。因此我们如果想要使用自己编译的内核，就必须先将它拷贝到/boot 目录下。需要拷贝的文件包括压缩的内核映象文件（如前所述的 zImage 或 bzImage）和内核符号表文件（System.map）。System.map 是 Linux 内核映象（zImage）中的符号表文件。在这个文件中记录了内核映象中的符号及其对应的地址。系统正常启动时并不会读这个符号表，一般只在调试中才用到此文件。如果没有拷贝最新的 System.map，系统可能会警告此文件不是最新的。这虽然不会影响正常运行，但最好还是在编译内核后拷贝这个文件以便于内核引导出错时调试内核。

```
#cp /usr/src/linux/arch/i386/boot/zImage /boot/zImage-2.4.18
#cp /usr/src/linux/ System.map /boot/System.map-2.4.18
#ln -sf /boot/System.map-2.4.18 /boot/System.map
```

如果使用 lilo 启动 Linux，则编辑/etc/lilo.conf 文件修改系统引导配置。lilo 是一个 Linux 的启动管理器，在机器启动硬件检测后被执行。这里我们以 lilo 为例通过增加一个配置项引导加载新内核，其他启动管理程序（例如 grub）的配置方法也与之类似。下面是一个 lilo.conf 的例子：

```
prompt
timeout=50
default=linux
boot=/dev/hda
```

```
map=/boot/map
install=/boot/boot.b
message=/boot/message
lba32

image=/boot/vmlinuz-2.4.7-10
        label=linux
        read-only
        root=/dev/hda1
image=/boot/zImage-2.4.18
        label=newkernel
        read-only
        root=/dev/hda1
```

其中最后四行是我们新增加的，image 指明了内核映象的存放位置，label 是 lilo 启动时供选择的提示关键字，root 是被加载（mount）为根目录（/）的磁盘分区。

这时，更改后的启动配置并没有写入 lilo，你必须执行 lilo 命令写入配置。

好了，现在你重启系统，lilo 就会出现你所加入的 newkernel 项，选择它将使用新的内核映象。

1.2　查看 Linux 内核状况

在使用一个操作系统时我们经常会想先了解一下系统状况，比如 CPU 的型号、内存的配置等硬件信息。另外，我们还希望监测内存使用情况等动态的系统信息，以便清楚掌握系统运行情况。

由上一节我们知道普通进程是运行在用户态下的，它们无法直接访问内核数据来了解系统信息。内核提供了系统调用作为访问这些数据结构的统一接口，但这种程序如果编写不当可能存在安全上的隐患。在现代的操作系统中实现了一个/proc 虚拟文件系统，通过它里面的一些文件，可以获取系统状态信息并且修改某些系统的配置信息。/proc 文件系统本身并不占用硬盘空间，它仅存在于内存之中，为操作系统本身和应用程序之间的通信提供了一个安全的界面。像 Linux 内核可卸载模块都在 /proc 文件系统中创建实体。当我们在内核中添加了新功能或设备驱动时，经常需要得到一些系统状态的信息，一般这样的功能可能需要经过一些像 ioctl()的系统调用来完成。系统调用界面对于一些功能性的信息可能是适合的，因为应用程序必须将这些信息读出后再做一定的处理。但对于一些实时性的系统信息，例如内存的使用状况，或者是驱动设备的统计资料等，我们更需要一个比较简单易用的界面来取得它们。/proc 文件系统就是这样的一个界面，我们可以简单地用 cat、strings 程序来查看这些信息。例如，查看系统内存的使用状况可以用如下的命令：

```
#cat /proc/meminfo
                total:      used:      free:      shared:     buffers:    cached:
Mem:            63627264    61816832   1810432      8192      696320      48074752
Swap:           67567616    9908224    57659392
MemTotal:       62136 kB
MemFree:        1768 kB
```

```
MemShared:          8 kB
Buffers:          680 kB
Cached:         43096 kB
SwapCached:      3852 kB
Active:          8056 kB
Inact_dirty:    36820 kB
Inact_clean:     2760 kB
Inact_target:     652 kB
HighTotal:          0 kB
HighFree:           0 kB
LowTotal:       62136 kB
LowFree:         1768 kB
SwapTotal:      65984 kB
SwapFree:       56308 kB
NrSwapPages:    14077 pages
```

下面就介绍从/proc 文件系统的各文件中能获取的信息。我们的目的就是将你领入到/proc 去探索系统状况，让你大概知道从系统中能得到什么信息以及从哪里可以得到。我们不可能详细列出每一文件的每一信息，而且不同版本在/proc 中的文件也有所区别（你会在后面学习如何在/proc 添加自己的文件），希望你能够自己去尝试，尤其是获得当前应用程序所需的信息。

1.2.1 系统信息

在/proc 文件系统根目录下有大量记录系统信息的文件和目录，表 1-1 列出了一些与进程无关的部分，目录和文件视内核配置情况而定，不一定都存在。用命令 procinfo 可以显示出由这些文件得到的大量系统信息。下面是运行 procinfo 的显示：

```
#procinfo
Linux 2.4.18-10 (bhcompile@stripples.devel.redhat.com) (gcc 2.96 20000731 ) #1 Thu Sep
6 17:27:27 EDT 2001 1CPU [exchange.(none)]

Memory:      Total        Used        Free      Shared     Buffers      Cached
Mem:         62136       58640        3496          12         836       34996
Swap:        65984        9320       56664

Bootup: Thu Mar 21 18:19:53 2002    Load average: 1.13 1.11 1.15 2/55 32305

user   :       1:35:18.04   3.1%  page in :  8061989  disk 1:   229457r   70958w
nice   :       0:00:05.58   0.0%  page out:  2040076
system:        1:20:43.46   2.7%  swap in :     5485
idle   :   1d 23:29:30.42  94.2%  swap out:     3355
uptime:    2d  2:25:37.50         context :  8306259

irq  0:  18153750 timer                 irq  8:          7 rtc
irq  1:      5022 keyboard              irq 10:    3320456 eth0
irq  2:         0 cascade [4]           irq 11:   13580885 eth1
```

```
irq  6:          3                          irq 14:     298799 ide0
irq  7:        248
```

表 1-1　/proc 根下文件和目录

文件/目录名	描述
apm	高级电源管理信息
bus	包含了总线以及总线上设备信息的目录，子目录以总线类型组织
cmdline	内核的命令行参数
cpuinfo	CPU 信息，包括主频、类型等信息
devices	系统字符和块设备编号及驱动程序名
dma	正在使用的 DMA 通道
driver	组织了不同的驱动程序
execdomains	和安全相关的 execdomain
fb	framebuffer 设备
filesystems	系统支持的文件系统类型
fs	文件系统需要的参数，对 NFS/export 有效
ide	包含了 IDE 子系统信息的目录
interrupts	系统注册的中断信息，其内容包括中断号、收到的中断数、驱动器名等
iomem	内存映象
ioports	I/O 端口使用情况
irq	与 CPU 有关的中断掩码
kcore	内核的 core 文件映象，记录了系统物理内存情况。可以使用 gdb 程序从中检查内核数据结构。该文件不是文本格式，不能用 cat 等文本查看器查看其内容
kmsg	内核消息，可以从该文件检索内核使用 printk()产生的消息。
ksyms	内核符号表，包括内核标识符地址和名称
loadavg	最近 1 分，5 分，15 分钟时候的系统平均负载量
locks	内核锁，记录与被打开的文件有关的锁信息
mdstat	被 md 设备驱动程序控制的 RAID 设备的信息
meminfo	内存信息
misc	杂项设备信息
modules	系统正在使用 module 信息
mounts	已经装载的文件系统
Net	保存网络信息的目录
partitions	硬盘分区情况
pci	PCI 总线上设备情况
scsi	SCSI 设备信息
slabinfo	slab 池信息
stat	静态统计信息，包括 CPU 的使用情况、磁盘、页面、交换、启动时间等数据
swap	交换分区的使用情况
sys	可以更改的内核数据的目录，其下包含的文件由表 1-2 说明
sysvipc	和 sys VIPC 相关数据文件目录，包括系统中消息队列（msg 文件）、信号量（sem 文件）、共享内存（shm 文件）的信息
tty	和终端相关数据
uptime	从系统启动到现在所经过的秒数及系统空闲时间
version	内核版本数据

/proc/sys 目录是一个特殊的目录，它支持直接使用文件系统的写操作，完成对内核中预定的一些变量的改变，从而达到更改系统特性的目的。比如说需要增加系统同时打开文件的个数，以提高使用

试读结束，需要全本PDF请购买 www.ertongbook.com