Linux

Programming Instances

网络编程实例

北京希望电子出版社 总监制

张 昆 编著

本版特点

- 理论与实践相结合
- 55个详尽的源码分析

适用范围

- Linux应用与开发的从业人员
- 大、专院校相关专业师生
- 社会初、中级培训班

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

内 容 简 介

本书通过 55 个精彩的实例，全面剖析了在 Linux 下编写网络应用程序的方法，并阐述了网络协议架构和开发规范。为了适应不同读者的需要，本书从最基本的 Linux 系统操作到网络技术的基本理念，逐步深入至 Linux/UNIX 下具体的编程实践，结合大量具体实例和编程经验，为读者展现 Linux 平台下网络编程的魅力。

全书由 13 章组成，内容涉及到 Linux 系统编程基础、TCP/UDP 协议、套接字编程概念及 I/O 模型、高级编程中需要用到的进程间通信同步、多路复用、多线程编程和一些高级套接字控制方法、IPv6 介绍以及网络安全等。本书最后还汇集了很多网络编程的具体实例，读者可以模仿书中的范例来开发自己的应用程序。

本书内容丰富，结构清晰，实例典型，文字简洁流畅，边讲边练。不但是 Linux 应用与开发的从业人员的指导书，而且也可作为大专院校相关专业师生教学与自学的参考书以及社会初、中级培训班教材。

读者在本书使用过程中的技术问题请与作者联系。E-mail：William-zhang@vip.sina.com。

本书源代码请读者在 www.b-xr.com 网址自由下载，文件名为 3786.zip。

系 列 书 名 ："十五"国家重点电子出版物规划项目·计算机知识普及和软件开发系列
专业程序员编程实作丛书（5）

书　　名 ：Linux网络编程教程

总 策 划 ：北京希望电子出版社

文本著作者 ：张威　编写

责 任 编 辑 ：杨如林

出版、发行者 ：北京希望电子出版社

地　　址 ：北京市海淀区知春路63号卫星大厦一层 100080

网址：www.bhp.com.cn

E-mail: lwm@bhp.com.cn

电 话：010-62520290,62521724,62528691,62630301,62524940,62521921,82610344

（发行） 010-62613322-215（门市） 010-82675588-501,82675588-201（编辑部）

经　　销 ：各地新华书店、软件连锁店

排　　版 ：希望图书输出中心　全卫

文本印刷者 ：北京双青印刷厂

开本 / 规格 ：787 毫米×1092 毫米　1/16　27 印张　628 千字

版次 / 印次 ：2002 年 6 月第 1 版　2002 年 6 月第 1 次印刷

印　　数 ：1-3000 册

本 版 号 ：ISBN 7-900101-24-1

定　　价 ：52.00 元

前　言

Linux 是一个完全免费的类 UNIX 操作系统，它承袭了 UNIX 系统优秀而强大的功能和极佳的稳定性，全世界各地有几十万自愿者正在为这个充满魅力的操作系统的发展贡献自己的才能。正是 Linux 的这种完全公开代码的体制以及它与 Internet 的紧密结合给予了它顽强的生命力和无限的发展空间。

随着 Internet 的发展和普及，网络编程开发应用在越来越多的方面，可以说没有网络也就没有信息时代。

所以 Linux 以其出色的性能和低廉的价格成为中低档服务器和大量个人 PC 的首选操作系统。了解、熟悉和掌握 Linux 系统及其环境下的编程无疑也变得越来越重要。基于 C 语言的 Linux 网络应用程序开发更是 Linux 平台编程的重要一环。

本书致力于阐述 Linux 平台下网络编程的理念和实际应用的技巧，旨在帮助更多的人领会到 Linux 网络开发的精髓。

本书读者对象

如果您是一个对编程刚刚入门的 Linux 爱好者，本书将是跨进 Linux 开发世界的一把钥匙，引领您从最基本的网络协议架构到最前沿的发展动态，向您展现 Linux 平台编程开发的魅力，使您成为一个经验丰富的网络程序员。

如果您是一个优秀的 windows 程序员，本书将深入浅出的引导您进入 Linux 的自由编程世界中。从点点滴滴的基本函数介绍到庞大详尽的程序源码分析，能使您在较短的时间里掌握 Linux 甚至 UNIX 网络编程，领会到 Linux/Unix 网络开发的乐趣。

如果您仅仅想了解网络架构和协议，本书也不失为一本很有价值的参考书，因为它从最基本的 TCP(UDP)/IP 架构到最新的 IPv6 都有详尽的解释与编程实例。

理论学习与实际编程应用的紧密结合是本书的特点，大量的实例、详尽的源码分析将更有利于概念的理解和动手能力的提高。

本书内容简介

本书着重论述了 Linux 平台下的网络编程，全书共分三部分：

基础篇：

本篇主要包括 Linux 平台开发环境的介绍、基本网络架构与 TCP/UDP 传输层协议的基础知识和套接字编程的概念。

首先，快速回顾了 Linux 平台的基本操作与基本编程，包括文件系统与操作、进程间通信，为您开发高级网络程序提供参考，同时对于不熟悉 Linux 系统的读者也可以提供 Linux 编程入门的指导；然后，又从基础的网络架构与概念出发，阐述网络运作及网络编程需要掌握的最基本知识，包括传输层协议 TCP/UDP 等等，并详细介绍了 TCP 套接字编程

和 UDP 编程，一些完善的 TCP 编程实例将有助于对这些概念的理解。

最后，介绍了套接字编程概念，包括套接字中的 I/O 模型和套接字属性控制的介绍。套接字编程的介绍是进行网络编程的基础。

提高篇：

高级网络编程需要涉及到进程间通信的同步、多路复用、多线程编程和一些套接字高级控制。对于 Linux 高级编程，进程间通讯、多路复用和信号驱动 I/O 是很重要的内容，其中的大量实例也可以作为实际编程参考；对于网络编程的高级开发，需要了解高级套接字 I/O 操作、多线程编程及其网络应用。

对于 IP 协议与属性，特别针对 IPv6，本书还详尽介绍了它的特点与编程应用；最后还介绍了有关网络安全等问题。

实例篇：

本书的特点就是将理论和应用实例相结合，每个具体的实例都可以实现一个相对完善的功能，对代码的详尽分析将有助于您理解一个完善的网络程序的编写过程。

本篇主要包括十一个实例，包括 ping 程序代码、聊天室的实现、端口扫描程序、网页更新检查程序、sniffer 的基本实现、IP 包检查程序、IP 欺骗实例、路由测试程序、Linux 防火墙的编写、守护进程和一个简单的 FTP 即 TFTP 协议代码。

附录中还介绍了如何编译和调试实例篇中的程序，它们可以作为您一般 Linux 开发的参考资料。

本书由白金刚、杜强、张威编著。

参加本书编写工作的有董毓、李强、焦璐、何永宁、徐振宇、宋丽、郝菲、赵彬、张艳刚、苗爱光、李娟、席新希、孟凯波、张磊、王静、梁靖娴、陶守坤、赵鹏、杨明、李家裕、张俊峰、刘森、俞燕云、李钊等。由于编者水平有限，不能做到面面俱到，加上时间仓促，书中难免出现一些纰漏，诚恳希望读者对本书提出批评建议。

作者

目　录

基 础 篇

提 高 篇

实 例 篇

附　录

基 础 篇

第 1 章 Linux 平台环境简单回顾

概述

本章将简要地介绍 Linux 操作系统的基本构成，包括其中 3 个基础部分：文件系统、I/O 操作以及进程控制和信号的概念。在这里将主要对一些具体函数进行介绍，目的是为了在以后的编程实践中理解和运用这些函数。

第一节就文件系统及其操作进行介绍，Linux 操作系统的特点决定了文件系统的总体结构和操作方式，它可以支持多种文件系统，本节将就一些特殊文件的功能、操作和一般性文件的操作函数进行介绍。

第二节介绍文件的标准 I/O 操作，包括对文件的读、写、错误输出、缓冲区操作等。然后介绍进程和信号，这些概念都是网络编程中的一些最基本的概念。

本章内容实际上只是作为 Linux 编程的一些最基本内容的回顾，相当于快速参考手册，这对于不熟悉 Linux 操作系统的读者也可以作为一个入门简介和预备知识。而对 Linux/UNIX 比较熟悉的读者可以跳过这一章。

1.1 文件系统及其操作

与其他操作系统一样，Linux 的文件系统也是目录和文件的一种层次安排。

1.1.1 文件系统结构

文件系统的总体结构：

Linux 支持多种文件系统，以便于它和其他的操作系统并存。这些文件系统包括 ext、ext2、xia、minix、umsdos、msdos、vfat、System V 等。一个典型的 Linux 文件系统结构如图 1-1 所示。

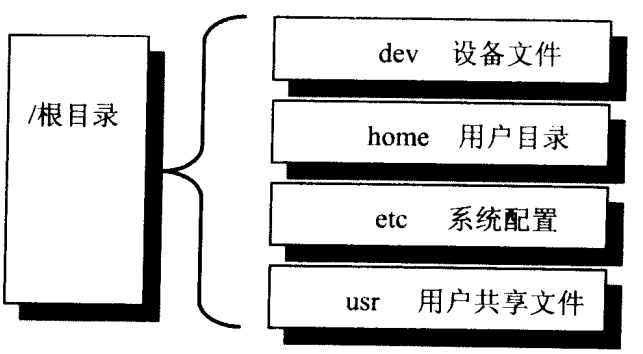

图 1-1 Linux 文件系统的结构

从初始的根目录（/）逐级到高层次的目录，每个目录都有自己的标识和属性，在逻辑

上可以认为每个目录项都包含一个目录文件名和说明该文件属性的信息，包括文件类型、文件长度、文件的所有权和使用许可权以及文件的最后修改时间等。

文件被链接在每个目录中，这个文件所在目录的具体位置就是该文件的路径。路径名若以根目录开头，则称为绝对目录；否则称为相对目录。每个文件也同样有以上的文件属性信息。

图 1-1 是从用户的角度看的 Linux 文件系统结构：

- /dev 目录是系统设备文件的存放位置，提供系统中各种设备的说明。
- /etc 中的文件提供系统中各种关于用户账号、网络等的配置信息。
- /usr 是用户共享文件目录，其中的/usr/include 存放各种库文件。
- /home 是用户账号的缺省目录，用户登录时默认进入这个目录。

从系统实现的角度看，文件系统的层次从高到低可以分为应用程序、系统调用、文件子系统、高速缓存、设备驱动和具体的硬件设备等，可以描述为图 1-2 所示。

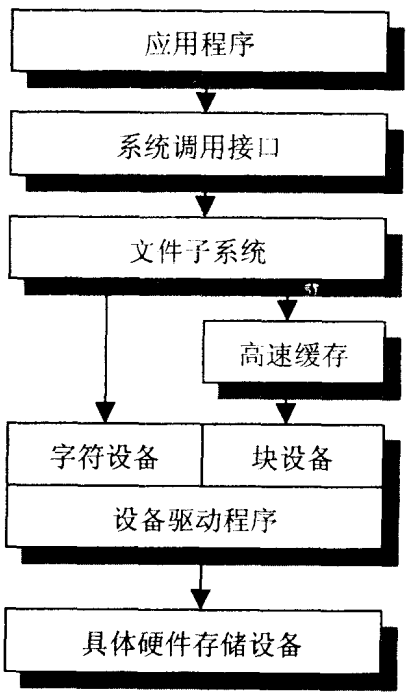

图 1-2　文件系统的操作层次

整个文件系统可以分为根文件系统和子文件系统。根文件系统是整个文件系统的基础，而子文件系统则是以目录树中的某个子目录的形式出现的。

文件子系统可以是多种文件系统，应用程序在对它进行操作的时候需要通过系统调用接口，这样不同的文件子系统就可以给应用程序一个统一的调用接口，Linux 就可以实现各种文件子系统的兼容。为了提高数据传送效率，文件子系统还需要通过高速缓冲机制作为中介，最后由设备驱动程序对硬件存储设备进行操作。

在 Linux 系统中要搞清楚文件连接的概念。我们用下面的图来描绘出磁盘、分区、文件系统以及详细的节点数据关系，如图 1-3 所示。

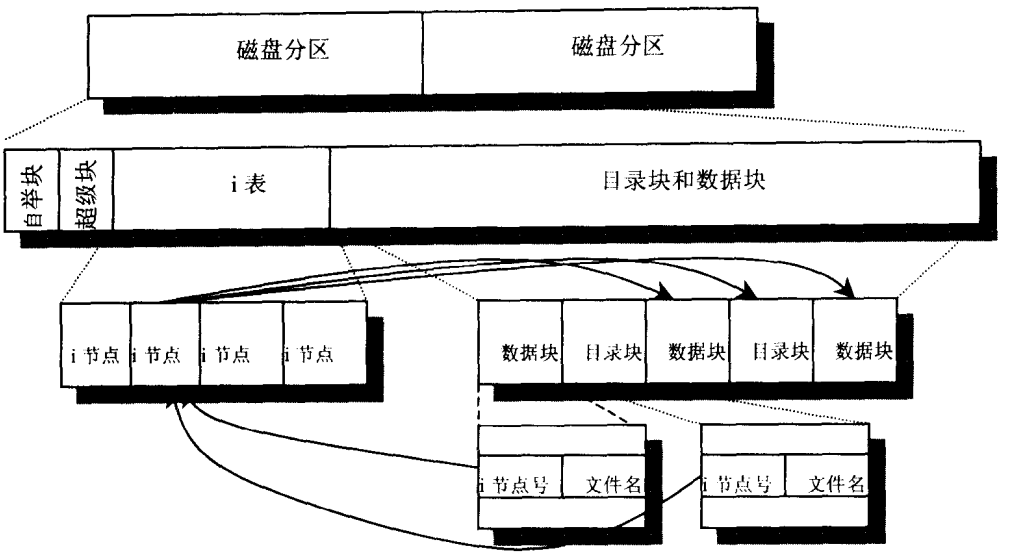

图 1-3 Linux 文件系统结构详图

每一个 i 节点都有一个连接计数，计算连接到这个 i 节点的目录块的数目。

所谓文件或目录的连接实际上就是磁盘上的数据被一个以上的文件名所引用，目录里存的 i 节点号指向分区中的 i 节点列的相应位置，而该处包含了除文件名外所有与文件有关的信息，如文件类型、存取权限、文件长度、数据指针等。目录项内容包括索引节点编号、目录项名称长度和目录项名称本身。用图 1-4 来详细表示目录项布局和 i 节点的关系。

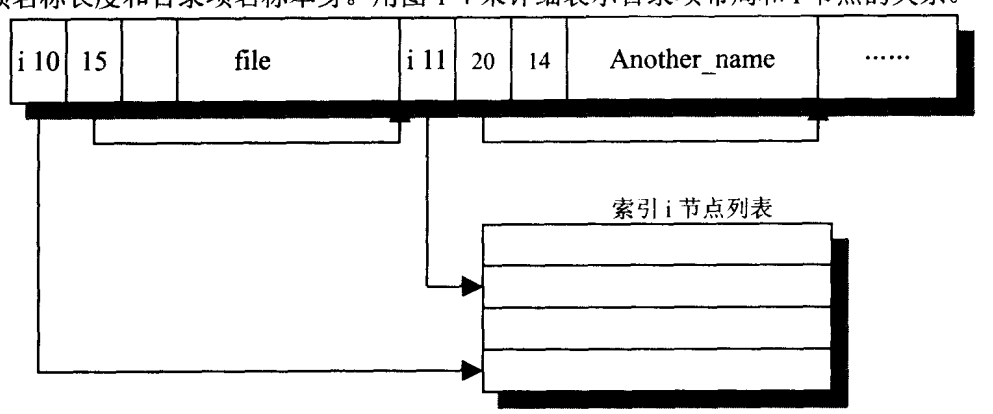

图 1-4 目录项布局

当我们执行删除一个文件时（rm 命令），实际上删除的是该文件的一个连接，直到连接计数减少到零以后才会从磁盘上释放该文件所占有的空间。

内核操作的实际上是 i 节点列表和其中的数据结构，这就相当于一个指针，指向实际数据块。

另外一种连接方式称为符号连接，对于这种文件，文件内容实际上是符号连接所指向的文件路径名。

1.1.2　文件 I/O 操作

本节所引用的函数通常被称之为不带缓存的 I/O（unbuffered I/O），这与 1.2 节的标准 I/O 操作是不同的。不带缓存指的是每个这样的函数如 read 和 write 都调用内核中的一个系统调用。

在涉及到多个进程以及它们之间的资源共享时我们将介绍原子操作的概念，并将说明 dup、fcntl 和 ioctl 函数。

对于内核而言，所有打开的文件都由一个非负整数描述，称为文件描述符。

下面简要的介绍一下文件 I/O 函数及其用法：

（1）open

此函数为打开或创建一个文件。

格式：

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open (const char *name, int oflag[|optional parameter][,mode]);
```

参数：

name 是打开或创建的文件名；

oflag 为 O_RDONLY 只读打开；

oflag 为 O_WRONLY 只写打开；

oflag 为 O_RDWR 读写打开；

可选参数：

O_APPEND 每次都添加到文件尾部；

O_TRUNC 只读或只写打开文件，写入前删除原有数据；

O_CREAT 文件不存在则创建（需要第三个参数 mode 来说明存取许可位）；

O_EXCL 如果文件已经存在而又指定了 O_CREAT，则报错；

O_NOCTTY 如果打开的文件为系统终端，则终端不可以作为正在操作 open()的控制终端。

Mode 是文件的访问许可权位：

S_IRWXU	00700	屏蔽用户操作
S_IRUSR (S_IREAD)	00400	用户读许可位
S_IWUSR (S_IWRITE)	00200	用户写许可位
S_IXUSR (S_IEXEC)	00100	用户执行许可位
S_IRWXG	00070	屏蔽组操作
S_IRGRP	00040	组读许可位
S_IWGRP	00020	组写许可位
S_IXGRP	00010	组执行许可位
S_IRWXO	00007	屏蔽其他操作

S_IROTH	00004	其他读许可位
S_IWOTH	00002	其他写许可位
S_IXOTH	00001	其他执行许可位

返回值：

成功则返回文件描述符；失败则返回-1。

（2）creat

此函数为创建一个新文件。

格式：

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int creat (const char *name, mode_t mode);
```

参数：

name 是创建文件名；

mode 是创建文件时的存取许可位。

这个函数可以用 open 函数来等效，它只能以只写方式打开文件，等于

```
open (name, O_WRONLY|O_CREAT|O_TRUNC,mode);
```

返回值：

成功则返回文件描述符；失败则返回-1。

（3）close

此函数为关闭一个已经打开的文件

格式：

```
#include <unistd.h>
int close (int file);
```

参数：

file 是已经打开的文件描述符。

返回值：

成功返回 0，否则返回-1。

（4）lseek

此函数为指定一个当前文件位移量。这个位移量是一个非负整数，用来说明从文件开始处计算的字节数。

格式：

```
#include <unistd.h>
#include <sys/types.h>
off_t lseek (int file, off_t offset, int whence);
```

参数：

file 是需要指定位移量的文件描述符；

offset 是位移量大小，是任意整数；

whence 是 SEEK_SET，则将位移量设置在距文件开始处 offset 字节的位置；

whence 是 SEEK_CUR，则将位移量设置在距当前值加 offset 字节的位置；
whence 是 SEEK_END，则将位移量设置为文件长度加 offset 字节的位置。
返回值：
成功返回文件位移；出错返回-1。

（5）read

此函数为从文件中读入数据。

格式：

```
#include <unistd.h>
size_t read (int file, void *buff, size_t bytes);
```

参数：

file 是目标文件的文件描述符；
buff 是输入输出缓冲区，指针指向要读入的缓冲区；
bytes 是每次读取的数据量。

返回：

返回读到的字节数，若已到尾部则返回 0，若出错返回-1。

（6）write

此函数为向文件中写入数据。

格式：

```
#include <unistd.h>
size_t write (int file, void *buff, size_t bytes);
```

参数：

file 是目标文件的文件描述符；
buff 是 I/O 缓冲区，指针指向要写入的缓冲区；
bytes 是每次写入的字节数；

返回值：

返回写入文件的字节总数，大小一定小于缓冲区容量。出错返回-1。

（7）dup

此函数可以用来复制一个现存的文件描述符。

格式：

```
#include <unistd.h>
int dup(int oldfile);
int dup2(int oldfile, ine newfile);
```

参数：

oldfile 是旧的文件描述符；
newfile 是新的文件描述符；
dup2 可以指定新的文件描述符；

返回值：

如果操作成功则返回新的文件描述符；失败返回-1。

（8）fcntl

fcntl 用途 1：复制描述符（cmd=F_DUPFD）。

fcntl 用途 2：获取/设置文件描述符标志（cmd=F_GETFD or F_SETFD）。惟一标志：FD_CLOEXEC。

fcntl 用途 3：获取/设置文件状态标志（cmd=F_GETFL or F_SETFL）。只能修改：O_APPEND、O_NONBLOCK 和 O_ASYNC。

fcntl 用途 4：获取/设置记录锁（cmd=F_GETLK, F_SETLK or F_SETLKW）。

fcntl 用途 5：获取/设置异步 I/O（cmd=F_GETOWN, F_SETOWN, F_GETSIG* or F_SETSIG*）。

格式：

```
#include <unistd.h>
#include <fcntl.h>
int fcntl (int fd, int cmd);
int fcntl (int fd, int cmd, long arg);
int fcntl (int fd, int cmd, struct flock *lock);
```

参数与返回值：

fd 是当前文件描述符；

arg 是一个整数，用于和 cmd 的配合使用；

*lock 是一个指向结构体的指针，用于和 cmd 的记录锁功能配合；

cmd 是一个指令，用它来实现不同的功能调用，当 cmd 为：

- F_DUPFD 复制现存的描述符到 arg，功能同 dup2，成功返回新的文件描述符，错误返回-1；
- F_GETFD 获得文件描述符标记，成功则返回该描述符标记，错误返回-1；
- F_SETFD 设置文件描述符标记为 arg 指定的值，成功返回该值，错误返回-1；
- F_GETFL 读取并返回已打开文件的状态标志，错误返回-1；
- F_SETFL 设置文件状态标志为 arg 的值，可以更改的几个标志是 O_APPEND，O_NONBLOCK，O_ASYNC，成功返回相应标志，错误返回-1。

F_GETLK，F_SETLK，F_SETLKW 为获得/设置记录锁。

（9）ioctl

此函数是 input/output control 的缩写，即输入输出控制。它可以通过一个文件描述符来控制字符设备。由于 Linux 将设备文件等特殊文件和一般文件平等对待，那么要对特殊文件作一些特殊操作就要用到 ioctl 参数。通常 ioctl 的最大使用方面是终端 I/O。

格式：

```
#include <sys/ioctl.h>
int ioctl (int file, int request, char *argp|struct termios st);
```

参数：

file 是一个已打开文件的描述符；

request 是一个编码代表的命令类型；

argp 是该命令的参数，termios 是一个在 termios.h 中定义的结构体，其中的参数可以告诉设备驱动程序如何处理输入输出数据。

返回值：

成功返回 0，出错返回-1。

1.1.3　文件、目录及操作

下面介绍一些有关文件和目录操作的函数。

1. 文件类型

Linux 下的文件类型可以分为普通文件、目录文件、管道、套接字、符号连接文件、字符特殊文件、块设备特殊文件等。

普通文件是最常见的文件类型，用于存储数据；目录文件并不包含数据信息，只是包含了其他文件信息和指针；特殊文件用于设备的管理；套接字文件用于进程间的通信，包括进程间通信和网络通信；符号连接文件指向另外一个文件。

在具体的文件操作中，经常会用到 stat()、fstat()和 lstat()函数。

stat()、fstat()和 lstat()函数

这些函数经常用来取得关于文件的一些信息。fstat 功能与 stat 类似，只是不以文件的路径名称作为标识，而是用文件描述符标识目标文件，获得该文件的有关信息。lstat 形式和功能都与 stat 很类似，但是当目标文件是符号连接文件时，lstat 返回该符号连接的有关信息。

格式：

```
#include <sys/stat.h>
#include <unistd.h>
int stat (const char *name, struct stat *buf);
int fstat(int filedes, struct stat *buf);
int lstat (const char *file, struct stat *buf);
```

参数：

name 是目标文件的完整路径和文件名；

filedes 是目标文件的文件描述符；

结构体 stat 是在/usr/include/bits/stat.h 中定义的，它说明了文件的一些信息结构；buf 就是用来存放目标文件信息结构的结构体变量。lstat 与 stat 不同的是当目标文件是符号连接时，返回的不是符号连接所指向文件的信息，而是该符号连接文件的有关信息。以下是结构体 stat 的定义：

```
struct stat {
        dev_t       st_dev;     /* device */
        unsigned short  __pad1;     /* padding */
        ino_t       st_ino;     /* inode
        umode_t     st_mode;    /* access mode */
        nlink_t     st_nlink;   /* number of hard links */
        uid_t       st_uid;     /* uid */
```