实战 Linux Socket 编程

Warren W.Gay 著

詹俊鹊 于卫 译

西安电子科技大学出版社

2002

内 容 简 介

目前已经有很多介绍计算机网络的书籍，但是它们之中的大多数似乎更适合于高级编程者，而对于众多只想了解使用方法的初学者而言，就显得太深奥了。

读者通过学习本书，可以掌握有关套接口编程的知识。同时，本书使用一种称为“by example”的方法来提高读者的学习效率。书中每一章的内容都是建立在前一章的基础之上的。第一部分“基本套接口概念”在阐明域和地址族、套接口的类型以及面向连接/非连接协议等基本概念的基础上，介绍了简单的客户/服务器程序的编写方法和主机名/网络名查询程序的编写方法。在掌握了第一部分“基本套接口概念”之后，读者就可以开始学习第二部分“高级套接口编程”，这对于有些读者而言可能是个挑战。这一部分介绍了套接口标准 I/O、并发客户服务程序、套接口选项、UDP 广播、带外数据、inetd 守护进程、网络安全程序设计以及信任状和文件描述符等较为深入的主题；并通过最后一章的应用实例，将前面介绍的诸多概念融合在一起。

图书在版编目(CIP)数据

实战 Linux Socket 编程 / Warren W.Gay 著；詹俊鹄，于卫译.

—西安：西安电子科技大学出版社，2001.12

ISBN 7-5606-1089-7

Ⅰ. 实…　Ⅱ. ① Gay… ② 詹… ③ 于…　Ⅲ. 网络—Linux 操作系统—程序设计

Ⅳ. TP393

中国版本图书馆 CIP 数据核字(2001)第 081422 号

责任编辑　毛红兵　龙晖

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　话　(029)8227828　邮　编　710071

http://www.xduph.com　E-mail: xdupfxb@pub.xaonline.com

经　销　新华书店

印　刷　西安文化彩印厂

版　次　2002 年 1 月第 1 版　2002 年 1 月第 1 次印刷

开　本　787 毫米×1092 毫米　1/16　印张 27.5

字　数　654 千字

印　数　1～4 000 册

定　价　36.00 元

ISBN 7-5606-1089-7 / TP・0541

XDUP 1360001-1

＊＊＊ 如有印装问题可调换 ＊＊＊

前 言

目前已经有很多介绍计算机网络的书籍，但是它们之中的大多数似乎更适合于高级编程者，而对于众多只想了解使用方法的初学者而言，就显得太深奥了。这就像没有必要让一个司机了解关于汽车的各种原理一样。

读者通过学习本书，可以掌握有关套接口编程的知识，这样，网络对于读者而言，就像一个可以随时打开开关并使用的电器一样简单。同时，本书使用一种称为“by example”的方法来提高读者的学习效率。书中每一章的内容都是建立在前一章的基础之上的，在掌握了第一部分“基本套接口概念”之后，读者就可以开始学习第二部分“高级套接口编程”，这对于有些读者而言可能是个挑战。最后一章是一个应用实例，它将前面介绍的诸多概念融合在一起。

by Example 系列

by Example 系列使用一种可能是最好的编程教学方法，通过“example”来教会读者如何进行编程。它就像一位指导者，帮助你纵览全局，并提供详尽的例程，向你展示使用新概念的方法。

by Example 系列的主导思想非常简单：通过大量的实例来学习计算机编程。命令说明、格式语法和语言参考都不足以教会一个新手如何掌握一门编程语言；只有尽快上手，运行并使用这些例程，学习者才可以获得更多的知识，而不仅仅是对语言的一种感觉。对于那些只掌握了一些基本概念的新手，如果他们在学习过程中能够踏踏实实地使用并研究这些例程，他们编程的技能和水平将在潜移默化中得到提高。

本书的读者对象

任何人，只要你想在 Linux 或 UNIX 平台上编写网络程序，都应该阅读本书。当然，为了给读者提供最有教育意义的经验，本书中所有例程都针对 Linux 平台，并进行了精心的修改。

本书的所有例程最好是运行于 Red Hat 6.0 以上版本或与之相当的 Linux 发布版本。因为命令 netstat(1)在近一段时间发生了很大的变化，所以如果在较老的 Linux 版本上运行例程可能会遇到问题；而且在老一些的 Linux 版本上，最好启动 /proc 文件系统，只有这样才能充分发挥例程的优势。

本书中的约定

为了读者理解上的方便，也为了将内容表述得更清楚明白，本书采用了一些通用的惯例。

注意：

“注意”栏中的内容用于澄清一些概念和过程，同时，它也扩充了每一章的内容。

提示：

“提示”栏中的内容为一些一般性的问题提供了简化的操作和解决方案。

警告：

“警告”栏中的内容用于提醒读者注意在 Linux 编程时可能会遇到的绊脚石。仔细阅读这些信息可以帮助你免除不少的烦恼，并节省大量宝贵的时间。

源代码的网址

请访问以下网址：

http://www.quecorp.com/series/by_example/

在那里你可以找到本书例程的源代码以及一些辅助资料。

目　录

第一部分　基本套接口概念

第一章　套接口简介 3

1.1　简要的历史回顾 3
1.2　理解套接口 4
1.2.1　定义套接口 4
1.2.2　使用套接口 5
1.2.3　引用套接口 6
1.3　套接口和管道 6
1.4　创建套接口 7
1.4.1　socketpair(2)使用范例 8
1.4.2　运行例程 10
1.5　用套接口实现 I/O 11
1.6　关闭套接口 15
1.6.1　shutdown(2)函数 15
1.6.2　关闭向套接口的写入 15
1.6.3　处理复制套接口 16
1.6.4　关闭从套接口的读入 17
1.6.5　什么时候不能使用 shutdown(2) 17
1.7　客户/服务器应用编程 18

第二章　域和地址族 25

2.1　无名套接口 25
2.2　域 25
2.3　生成套接口地址 26
2.4　生成本地地址 27
2.4.1　生成传统本地地址 28
2.4.2　生成抽象本地地址 32
2.5　生成 Internet(IPv4)套接口地址 35
2.5.1　理解网络字节序 36
2.5.2　在大端/小端字节序之间实现转换 37
2.5.3　初始化一个通配的 Internet 地址 38
2.5.4　初始化一个特定的 Internet 地址 39
2.6　生成 X.25 地址 41
2.7　生成其他地址族 43
2.8　AF_UNSPEC 地址族 47

第三章　地址转换函数 48

3.1　Internet IP 地址 48
3.1.1　Internet 地址分类 48
3.1.2　理解网络掩码 49
3.2　分配 IP 地址 53
3.2.1　私有 IP 地址 53
3.2.2　保留 IP 地址 54
3.3　操作 IP 地址 54
3.3.1　inet_addr(3)函数 54
3.3.2　inet_aton(3)函数 58
3.3.3　inet_ntoa(3)函数 60
3.3.4　inet_network(3)函数 62
3.3.5　inet_lnaof(3)函数 64
3.3.6　inet_netof(3)函数 65
3.3.7　inet_makeaddr(3)函数 66

第四章　套接口的类型与协议 70

4.1　指定套接口的域 70
4.1.1　PF_INET 和 AF_INET 71
4.1.2　使用 PF_LOCAL 和 AF_LOCAL 71
4.2　使用 socket(2)函数 72
4.3　选择套接口类型 72
4.3.1　理解 SOCK_STREAM 套接口类型 73
4.3.2　理解 SOCK_DGRAM 套接口类型 74
4.3.3　理解 SOCK_SEQPACKET 套接口类型 75

4.4 选择协议 75
4.4.1 使用 PF_LOCAL 和 SOCK_STREAM 76
4.4.2 使用 PF_LOCAL 和 SOCK_DGRAM 77
4.4.3 使用 PF_INET 和 SOCK_STREAM 77
4.4.4 使用 PF_INET 和 SOCK_DGRAM 78
4.5 套接口 Domain 和 Type 参数的总结 79
4.6 Linux 支持的其他协议 80

第五章 为套接口绑定地址 86

5.1 bind(2)函数的作用 86
5.2 使用 bind(2)函数 86
5.3 获得套接口地址 89
5.3.1 编写函数 sock_addr() 90
5.3.2 获得对等套接口地址 94
5.4 接口与定址 96
5.4.1 指定接口地址的图例 97
5.4.2 绑定一个特定的接口地址 97
5.4.3 绑定通配接口 98

第六章 面向非连接的协议 100

6.1 通信的方法 100
6.1.1 非连接通信的优点 100
6.1.2 非连接通信的缺点 101
6.2 实现数据报的输入和输出 101
6.2.1 sendto(2)函数介绍 102
6.2.2 recvfrom(2)函数介绍 103
6.3 编写 UDP 数据报服务器程序 104
6.4 编写 UDP 数据报客户程序 110
6.5 测试 UDP 数据报服务器/客户程序 114
6.5.1 在没有服务器的状态下进行测试 115
6.5.2 使用非缺省的 IP 地址进行测试 116
6.5.3 在客户程序中省略 bind(2)调用 117
6.5.4 对通配地址的应答 117

第七章 面向连接的协议——客户端 119

7.1 通信方法的回顾 119
7.2 Internet 服务 120
7.2.1 /etc/services 文件 121
7.2.2 使用 getservent(3)函数 121
7.2.3 使用 setservent(3)函数 124
7.2.4 使用 endservent(3)函数 125
7.2.5 通过名字和协议查询服务 125
7.2.6 通过端口和协议查询服务 126
7.3 /etc/protocols 文件 126
7.3.1 使用 setprotoent(3)函数 129
7.3.2 使用 endprotoent(3)函数 129
7.3.3 通过名字查询协议 129
7.3.4 通过协议号查询协议 130
7.4 编写 TCP/IP 客户程序 130
7.4.1 connect(2)函数 131
7.4.2 为编写客户程序做准备 131
7.4.3 daytime 客户程序 133
7.5 在 SOCK_DGRAM 套接口中使用 connect(2) 136

第八章 面向连接的协议——服务器端 138

8.1 服务器的作用 138
8.2 listen(2)函数 139
8.2.1 监听队列 140
8.2.2 指定 backlog 的值 140
8.3 accept(2) 函数 141
8.4 编写一个 TCP/IP 服务器程序 143
8.5 修改客户程序 150

第九章 主机名和网络名查询 154

9.1 理解名字的必要性 154
9.2 uname(2) 函数 155
9.3 获取主机名和域名 157
9.3.1 gethostname(2)函数 157
9.3.2 getdomainname(2)函数 158
9.3.3 测试 gethostname(2)函数和 getdomainname(2)函数 158
9.4 解析远程地址 160
9.4.1 错误报告 160
9.4.2 报告 h_errno 错误 161

9.4.3 gethostbyname(3)函数 161
9.4.4 gethostbyname(3)函数应用示例 163
9.4.5 gethostbyaddr(3)函数 166
9.4.6 sethostent(3)函数 172
9.4.7 endhostent(3)函数 173

第二部分 高级套接口编程

第十章 套接口上的标准 I/O 177

10.1 使用标准 I/O 流的必要性 177
10.2 连接套接口与流 178
10.3 关闭套接口流 179
10.4 分开使用读写流 179
10.4.1 复制套接口 180
10.4.2 关闭套接口上的读/写流 181
10.5 建立通信连接 181
10.5.1 只关闭写端 182
10.5.2 只关闭读端 182
10.5.3 同时关闭读写端 183
10.6 中断处理 183
10.7 定义缓冲操作 185
10.8 在套接口上使用 FILE 流 187
10.8.1 mkaddr( )函数 187
10.8.2 RPN 计算器引擎代码 192
10.8.3 测试 RPN 服务器程序 208

第十一章 并发客户服务程序 211

11.1 理解多客户问题 211
11.2 服务器函数概览 212
11.3 使用 fork(2)函数实现多客户服务 216
11.3.1 理解全局服务进程 221
11.3.2 理解子服务进程流 221
11.3.3 理解进程的终止处理 222
11.4 使用 select(2)函数设计服务器程序 223
11.4.1 select(2)函数简介 223
11.4.2 使用文件描述符集合 224
11.5 服务器程序设计中使用 select(2)函数 226
11.5.1 使用 select(2)函数的服务器程序 237
11.5.2 例程中的有关限制 239

第十二章 套接口选项 240

12.1 取套接口的选项值 240
12.2 设置套接口选项 244
12.3 取套接口类型（SO_TYPE） 249
12.4 设置 SO_REUSEADDR 选项 251
12.5 设置 SO_LINGER 选项 252
12.6 设置 SO_KEEPALIVE 选项 255
12.7 设置 SO_BROADCAST 选项 256
12.8 设置 SO_OOBINLINE 选项 257
12.9 选项 SO_PASSCRED 和 SO_PEERCRED 257

第十三章 UDP 广播 259

13.1 理解广播地址 259
13.1.1 在 255.255.255.255 上广播 260
13.1.2 增强 mkaddr.c 子程序的功能 260
13.2 服务器广播 261
13.3 接收广播 267
13.4 广播演示 271
13.5 面向网络的广播 272
13.5.1 启动广播 272
13.5.2 接收本地广播信息 273
13.5.3 接收远程广播信息 273
13.5.4 调试 274

第十四章 带外数据 275

14.1 带外数据概念 275
14.2 带外数据的必要性 276
14.3 套接口与带外数据 276
14.4 实现中的两种语义解释 276
14.5 使用带外数据 277

14.5.1 写带外数据 278
14.5.2 读带外数据 278
14.5.3 理解 SIGURG 信号 279
14.5.4 支撑子程序 280
14.5.5 使用 SIGURG 信号接收带外数据 284
14.5.6 发送带外数据 287
14.5.7 测试 oobrecv 和 oobsend 程序 289
14.6 紧急指针 290
14.6.1 TCP 紧急模式 290
14.6.2 tcp_stdurg=1 时的紧急模式 292
14.7 接收内嵌带外数据 293
14.7.1 确定紧急指针位置 293
14.7.2 使用内嵌带外数据 294
14.8 紧急指针的有关限制 298

第十五章 使用守护进程 inetd 301

15.1 一般服务程序所遵循的通用模式 301
15.2 inetd 简介 302
15.2.1 /etc/inetd.conf 配置文件 302
15.2.2 inetd 服务程序的设计参数 304
15.3 一个简单的 TCP 服务程序 305
15.3.1 通过 inetd 调用服务程序 307
15.3.2 禁止新增加的服务 310
15.4 数据报服务程序 310

第十六章 网络安全程序设计 312

16.1 什么是安全性 312
16.2 来自安全方面的挑战 312
16.3 区分合法用户与不合法用户 314
16.3.1 通过主机名或域名识别客户 314
16.3.2 通过 IP 地址识别客户 315
16.4 给 inetd 服务增加安全措施 316
16.4.1 集中式网络安全策略 316
16.4.2 理解 TCP 外包器概念 317
16.4.3 辨别客户的访问权限 318
16.5 安装外包器和服务程序 319
16.5.1 服务程序与外包器的日志代码 319
16.5.2 UDP 服务程序代码 321
16.5.3 一个简单的 TCP 外包器程序 326
16.6 客户端程序 329
16.7 安装并测试外包器 334
16.7.1 监视日志文件 335
16.7.2 启动 inetd 守护进程 335
16.7.3 测试外包器程序 336
16.7.4 测试服务器超时 337
16.7.5 卸载示范程序 337
16.7.6 数据报的缺陷 338

第十七章 传递信任状和文件描述符 340

17.1 问题 340
17.2 辅助数据简介 341
17.3 I/O 向量简介 341
17.3.1 I/O 向量（struct iovec） 342
17.3.2 readv(2) 函数和 writev(2) 函数 342
17.4 sendmsg(2)函数和 recvmsg(2)函数 344
17.4.1 sendmsg(2) 函数 344
17.4.2 recvmsg(2)函数 344
17.4.3 msghdr 结构 345
17.5 辅助数据结构和宏 346
17.5.1 cmsghdr 结构简介 346
17.5.2 cmsg(3) 宏简介 348
17.5.3 创建辅助数据 350
17.6 辅助数据例程 351
17.6.1 通用头文件 common.h 352
17.6.2 misc.c 模块 353
17.6.3 recvcred.c 模块 354
17.6.4 一个简单的 Web 服务器例程 web80 357
17.6.5 reqport( )函数 361
17.6.6 recv_fd( )函数 363
17.6.7 服务器程序 sockserv 366
17.6.8 send_fd( )函数 374
17.7 测试套接口服务器 377

第十八章 一个实用的网络工程项目 380

18.1 问题 380
18.2 解决报价服务问题 380
18.3 测试报价服务器程序 383

18.4 通过 get_tickinfo()获得报价单 390
18.5 通过 broadcast ( ) 函数进行报价广播 ... 398
18.6 分析客户端程序 400
18.7 编译并运行演示程序 405
18.7.1 启动报价服务器 qserve 406
18.7.2 启动客户端程序 mktwatch 407
18.7.3 如果 finance.yahoo.com 服务发生变化 408

附录 A 套接口函数快速索引 409

A.1 特定的套接口函数 409
A.2 套接口定址函数 410
A.3 读套接口函数 420
A.4 写套接口函数 412
A.5 其他套接口 I/O 函数 414
A.6 套接口控制函数 414
A.7 网络支持函数 416
A.8 标准 I/O 支持函数 418
A.9 主机名支持函数 418

附录 B 套接口相关的数据结构索引 420

B.1 套接口地址结构 420
B.2 其他数据结构 422
B.3 与 I/O 相关的数据结构 424

附录 C 一些常用的表格 425

附录 D 术语表 427

第一部分　基本套接口概念

套接口简介

域和地址族

地址转换函数

套接口的类型与协议

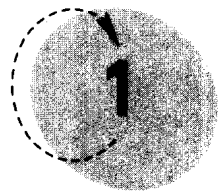
为套接口绑定地址

面向非连接的协议

面向连接的协议——客户端

面向连接的协议——服务器端

主机名和网络名查询



第一章 套接口简介

1957 年 10 月 4 日，星期五，苏联发射了世界上第一颗人造地球卫星——Sputnik，这标志着人类外太空时代的开始。这颗卫星有篮球那么大，在发射 98 分钟后到达运转轨道，任何无线电爱好者都可以通过短波 40.002 MHz 收听到它从头顶划过的声音。但当时谁又能想到，Sputnik 的升空竟然促使了 TCP/IP 和 Internet 的出现。

在本章中，我们将介绍以下内容：

- socket 发展的简要历史。
- socket 基础知识。
- Linux 内核和应用程序如何引用 socket。
- 一个用 C 程序语言编写的 socket 简单例程。

1.1 简要的历史回顾

针对 Sputnik 所带来的威胁，在美国总统艾森豪威尔的积极推动下，美国国会于 1958 年 1 月 7 日拨款成立 ARPA(the Advanced Research Projects Agency)。由于美国各政府部门在每次采购计算机时，都被要求从不同的设备制造商处购买，以保持公平公正，因此，ARPA 很快就发现，他们拥有的计算机难以互相兼容。1962 年，J.C.R.Licklider 提出了一种思想：即使计算机高度自治，它们也应该能够彼此互相通信。

在整个 60 年代，ARPA 都是在一群天才的指引下发展着。ARPA 网成为我们现在所熟知的 Internet 的前身。最终，ARPA 被并入 DARPA(the Defense Advanced Research Projects Agency)。

在 ARPA 网发展的同时，UNIX 也在迅速发展。加州大学 Berkeley 分校开发出了具有自己风格的 UNIX，我们通常称之为 BSD。由于 DARPA 想把自己从网络商业中剥离出来，所以它于 1979 年向加州大学 Berkeley 分校投资，以进一步发展 ARPA 网。1982 年，在系统内实现了 TCP/IP 协议的 4.1BSD 和 4.2BSD 被相继开发出来。我们在本书中将要了解的有关套接口和接口的概念就是基于 BSD 的。

Linux 充分利用了这笔丰厚的遗产，本书将详细介绍 Linux 对于 BSD 套接口的实现。

图 1.1 为这段历史的时间线，读者可能注意到，BSD 套接口是在 ARPA 出现 24 年后才发展起来的。

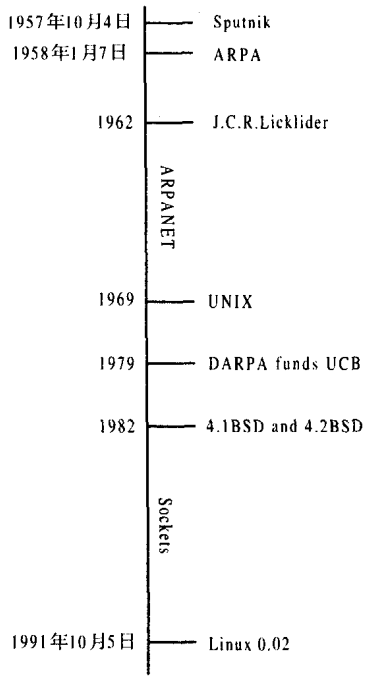


图 1.1 从 Sputnik 到 Linux

1.2 理解套接口

在使用套接口之前，应当先了解与其相关的若干概念。本节将简要介绍这些概念。

1.2.1 定义套接口

在用电话与他人联系时，你必须拿起听筒，拨叫对方的电话号码，然后等待对方的应答；当双方进行通话的时候，就建立了一个具有两个端点的通信线路，这两个端点是：

- 你的电话(在你的位置处)。
- 对方的电话 (在对方的位置处)。

双方的通信与通信的两个端点和它们之间的通信线路有关。图 1.2 就是用两个电话作为端点，并通过电话网互相连接的示例。

图 1.2 电话网的通信

Linux 中的套接口与电话非常相似。套接口代表通信线路中的端点，而端点之间就是数据通信网络。

套接口与电话的相似性还表现在另一方面。当你给某人打电话时，你拨叫的是对方用户的电话号码。而套接口中的网络地址就相当于电话号码。通过在程序中指定远程套接口的地址，就可以建立从本地套接口到远程套接口的通信。关于套接口地址，我们将在第二章“域和地址族”中讨论。

实际上，一个套接口仅仅是通信中的一个端点。Linux 提供了很多函数调用来操作套接口，而所有这些函数，我们都将在本书中进行详细的介绍。

1.2.2 使用套接口

由于套接口需要一整套专用的函数来操作，因此读者可能会认为套接口非常特殊。的确不错，套接口确实有一些特殊的地方，但实际上，它同我们已经很熟悉的文件描述符非常相似。

注意：

任何对函数名的引用，例如 pipe(2)，都意味着在你的 Linux 系统中，应该有该函数的在线文档(man page)。例如，我们要阅读 pipe(2)的信息，就可以键入命令：

```
$ man 2 pipe
```

这里的 2 代表手册的节代码。虽然节代码经常是可以省略的，但为了得到正确的信息，应该予以指定。

例如，使用 open(2)函数打开一个文件，如果调用成功，Linux 系统将返回一个文件描述符。接下来，程序就可以使用这个文件描述符对已打开的文件进行 read(2)、write(2)、lseek(2)和 close(2)等操作。

与上相似，套接口创建后，就如同一个文件描述符。我们可以使用同样的文件 I/O 函数，来对它进行读、写和关闭操作。在第十五章“使用守护进程 inetd”中，我们将学习使用套接口作为标准输入(文件描述符 0)、标准输出(文件描述符 1)、标准错误(文件描述符 2)。

注意：

与引用一个已打开的文件一样，套接口也是通过文件描述符来引用的，而且二者的文件描述符共享同一个“数字空间”——比如说，不可能在同一时间既打开一个描述符为 4 的套接口，又打开一个描述符为 4 的文件。

不过，在套接口和已打开的文件之间还是存在着以下差别：

- 不能在套接口上调用函数 lseek(2)(这限制也适用于管道)。
- 套接口可以和网络地址关联，文件和管道却不能。
- 套接口具有很多能够通过 ioctl(2)进行查询和设置的选项。
- 套接口必须在正确的状态下才能实现输入和输出，而已打开文件在任何时候都可以进行读或写操作。

1.2.3 引用套接口

调用 open(2)打开一个新文件，Linux 内核就会返回一个文件描述符，这个文件描述符是系统当前可用的，并且是数值最小的描述符，它或者是 0，或者是一个正整数。在有些系统中，文件描述符也称为句柄。

注意：

当需要新的文件描述符时，内核会返回当前可用的并且是数值最小的描述符。例如，如果关闭了标准输入(文件描述符 0)，紧接着又成功地打开了一个新文件，那么这个新文件的描述符就为 0。

现在假设程序已经打开了标准输入文件(文件描述符 0)、标准输出文件(文件描述符 1)和标准错误文件(文件 2)，然后实现下述操作(请注意内核是如何分配文件描述符的)：

(1) 调用 open(2)函数打开一个文件。

(2) 内核返回文件描述符 3，并指向打开的文件。这是因为 3 是目前尚未使用的而且是最小的描述符。

(3) 使用适当的函数调用(将在下文介绍)创建一个新套接口。

(4) 基于与(2)同样的理由，内核为该套接口返回文件描述符 4。

(5) 再次调用 open (2)，打开另外一个文件。

(6) 内核返回文件描述符 5，并指向该文件。

注意：Linux 内核在分配文件描述符时，并不区分这个描述符是分配给套接口，还是分配给一个已打开的文件。对程序员来说，这就意味着使用套接口与使用已打开文件是一样的。因为能够通过文件描述符来交替地引用文件和套接口，编程的灵活性就大大地提高了，也意味着使用 read (2)和 write (2)这样的函数，既可以操作文件，也可以操作套接口。

1.3 套接口和管道

在介绍套接口函数之前，我们先回顾一下 pipe(2)函数调用，看一看它所返回的文件描述符与套接口有什么不同。以下是 man 手册中 pipe(2)的语法：

```
#include <unistd.h>

int pipe(int filedes[2]);
```

如果 pipe(2)调用成功，内核将返回两个文件描述符。数组元素 filedes[0]代表管道读出端的文件描述符，数组元素 filedes[1]代表管道写入端的文件描述符。

为管道的两端各分配一个文件描述符，这样的安排方式具有一定的暗示作用。然而到底它与套接口的区别在哪里呢？其区别就在于 pipe(2)生成的是单向的通信通道。信息只能

从描述符 filedes[0]读出，也只能从 filedes[1]写进。任何反方向的操作只能导致 Linux 内核向程序报告错误。

套接口则不同，它允许进程进行双向通信。例如，一个进程使用描述符为 3 的套接口向远程进程发送数据，同时它也可以使用这个描述符为 3 的套接口接收来自远程进程的数据。这就是它与管道的根本差异。

1.4 创建套接口

在本节中，读者将看到，建立一个套接口就像建立一个管道那样容易，尽管可能涉及较多的函数参数。

以下是函数 socketpair (2)的语法：

```
#include <sys/types.h>
#include <sys/socket.h>

int socketpair ( int domain, int type, int protocol, int sv[2] );
```

头文件 sys/types.h 对于定义某些 C 的宏是必需的。

头文件 sys/socket.h 对于定义 socketpair(2)的函数原型是必需的。

函数 socketpair(2)需要四个参数，它们分别是

- 套接口的域名(int domain)。
- 套接口的类型(int type)。
- 使用的协议(int protocol)。
- 指向接收用于引用套接口文件描述符数组的指针 (int sv[2])

关于 domain 参数的解释将在第二章中说明。为了使用 socketpair(2)，通常为之提供值为 AF_LOCAL 的一个宏。

type 参数指明要生成的套接口类型。有以下两个值可供选择：

- SOCK_STREAM。
- SOCK_DGRAM。

关于这两个参数值，将在第四章“套接口的类型与协议”中说明。在本章中，我们只是简单地用 SOCK_STREAM 作为参数。

对 socketpair(2)函数来说，protocol 参数的值必须为 0。

参数 sv[2]是包含两个整型值的数组，每个整型值代表一个套接口，这个套接口类似于管道中某一端的端点。

如果函数调用成功，则返回值为 0；否则返回值为-1，表示调用失败，这时可以根据 errno 的值来判断错误的原因。

警告：

一定要测试函数的返回值以判断函数调用的成功与否。errno 是一个全局变量，当函数发生错误时，它将被置成一个指示错误类型的正数。errno 的值只在函数发生错误时设置。如果函数不发生错误，errno 的值就无定义，并不会被置为 0。

1.4.1 socketpair(2)使用范例

为了演示如何使用 socketpair(2)，下面提供了清单 1.1，读者可以在自己的 Linux 系统上进行测试。

警告：

手工键入本书中的例程时，不要键入每行左端的行号。行号只是为便于阅读和说明而添加的。

```
1:  /* 清单1.1 :
2:   *
3:   * socketpair(2) 函数示例：
4:   */
5:  #include <stdio.h>
6:  #include <stdlib.h>
7:  #include <unistd.h>
8:  #include <errno.h>
9:  #include <string.h>
10: #include <sys/types.h>
11: #include <sys/socket.h>
12:
13: int
14: main(int argc,char **argv) {
15:      int z;                          /* 返回的状态码 */
16:      int s[2];                       /* 套接口对 */
17:
18:      /*
19:       * 生成本地套接口对：
20:       */
21:      z = socketpair(AF_LOCAL,SOCK_STREAM,0,s);
22:
23:      if ( z == -1 ) {
24:           fprintf(stderr,
```