PEARSON

UNIX® Network Programming

Volume 2: Interprocess Communications, Second Edition

UNIX网络编程

卷2：进程间通信

（第2版·英文版）

[美] W. Richard Stevens 著

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

UNIX® Network Programming

Volume 2: Interprocess Communications, Second Edition

UNIX网络编程

卷2：进程间通信

（第2版·英文版）

[美] W. Richard Stevens 著

人民邮电出版社

北 京

图书在版编目（CIP）数据

UNIX网络编程. 第2卷, 进程间通信 : 第2版 = UNIX Network Programming, Volume 2: Interprocess Communications, Second Edition : 英文 / (美) 史蒂文斯 (Stevens,W.R.) 著. -- 2版. -- 北京 : 人民邮电出版社, 2016.1
ISBN 978-7-115-40131-1

Ⅰ. ①U… Ⅱ. ①史… Ⅲ. ①UNIX操作系统－程序设计－英文 Ⅳ. ①TP316.81

中国版本图书馆CIP数据核字(2016)第008174号

内 容 提 要

本书是一部 UNIX 网络编程的经典之作。进程间通信（IPC）几乎是所有 UNIX 程序性能的关键，理解 IPC 也是理解如何开发不同主机间网络应用程序的必要条件。本书从对 Posix IPC 和 System V IPC 的内部结构开始讨论，全面深入地介绍了 4 种 IPC 形式：消息传递（管道、FIFO、消息队列）、同步（互斥锁、条件变量、读写锁、文件与记录锁、信号量）、共享内存（匿名共享内存、具名共享内存）及远程过程调用（Solaris 门、Sun RPC）。附录中给出了测量各种 IPC 形式性能的方法。

本书内容详尽且具权威性，几乎每章都提供精选的习题，并提供了部分习题的答案，是网络研究和开发人员理想的参考书。

◆ 著　［美］W. Richard Stevens
责任编辑　杨海玲
责任印制　焦志炜

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
三河市海波印务有限公司印刷

◆ 开本：800×1000　1/16
印张：34.5
字数：672 千字　2016 年 2 月第 2 版
印数：1- 2 000 册　2016 年 2 月河北第 1 次印刷
著作权合同登记号　图字：01-2009-5714 号

定价：89.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

版 权 声 明

前　言

概述

大多数重要的程序都涉及进程间通信（Interprocess Communication, IPC）。这是受下述设计原则影响的自然结果：把应用程序设计为一组相互通信的小片断比将其设计为单个庞大的程序更好。从历史角度看，应用程序有如下几种构建方法。

(1) 用一个庞大的程序完成全部工作。程序的各部分可以实现为函数，函数之间通过参数、返回值和全局变量来交换信息。

(2) 使用多个程序，程序之间用某种形式的 IPC 进行通信。许多标准的 Unix 工具都是按这种风格设计的，它们使用 shell 管道（IPC 的一种形式）在程序之间传递信息。

(3) 使用一个包含多个线程的程序，线程之间使用某种 IPC。这里仍然使用术语 IPC，尽管通信是在线程之间而不是在进程之间进行的。

还可以把后两种设计形式结合起来：用多个进程来实现，其中每个进程包含几个线程。在这种情况下，进程内部的线程之间可以通信，不同的进程之间也可以通信。

上面讲述了可以把完成给定任务所需的工作分到多个进程中，或许还可以进一步分到进程内的多个线程中。在包含多个处理器（CPU）的系统中，多个进程也许可以（在不同的 CPU 上）同时运行，或许给定进程内的多个线程也能同时运行。因此，把任务分到多个进程或线程中有望减少完成指定任务的时间。

本书详细描述了以下 4 种不同的 IPC 形式：

(1) 消息传递（管道、FIFO 和消息队列）；

(2) 同步（互斥量、条件变量、读写锁、文件和记录锁、信号量）；

(3) 共享内存（匿名的和具名的）；

(4) 远程过程调用（Solaris 门和 Sun RPC）。

本书不讨论如何编写通过计算机网络通信的程序。这种通信通常涉及使用 TCP/IP 协议族的套接字 API，相关主题在第 1 卷[Stevens 1998]中有详细讨论。

有人可能会提出质疑：不应该使用单主机或非网络 IPC（本卷的主题），所有程序都应该在网络上的多台主机上同时运行。但在日常实践中，单主机 IPC 往往比网络通信快得多，而且有时还简单些。共享内存、同步等方法通常也只能用于单主机，跨网络时可能无法使用。经验和历史表明，非网络 IPC（本卷）与跨网络 IPC（第 1 卷）都是需要的。

本卷建立在第 1 卷和我写的另外 4 本书的基础上，这 5 本书在本书中简记如下：

- UNPv1：*UNIX Network Programming, Volume 1* [Stevens 1998]；
- APUE：*Advanced Programming in the UNIX Environment* [Stevens 1992]；
- TCPv1：*TCP/IP Illustrated, Volume 1* [Stevens 1994]；
- TCPv2：*TCP/IP Illustrated, Volume 2* [Wright and Stevens 1995]；
- TCPv3：*TCP/IP Illustrated, Volume 3* [Stevens 1996]。

在一本以“网络编程”为书名一部分的书中讨论 IPC 看似有点奇怪，但事实上 IPC 经常用于网络应用程序。我在《UNIX 网络编程》1990 年版的前言里就指出：“想知道如何为网络开发软件，必须先理解进程间通信（IPC）。”

与第1版的区别

本书完全重写并扩充了 1990 年版《UNIX 网络编程》的第 3 章和第 18 章。字数统计表明，现在的内容是第 1 版的 5 倍。新版的主要改动归纳如下。

- 不仅讨论了“System V IPC”的三种形式（消息队列、信号量以及共享内存），还对实现了这些 IPC 的新的 Posix 函数进行了介绍。（1.7 节将详细介绍 Posix 标准族。）我认为使用 Posix IPC 函数是大势所趋，因为它们比 System V 中的相应部分更具优势。
- 讨论了用于同步的 Posix 函数：互斥锁、条件变量以及读写锁。它们可用于线程或进程的同步，而且往往在访问共享内存时使用。
- 本卷假定使用 Posix 线程环境（称为“Pthreads”），许多示例都是用多线程而不是多进程构建的。
- 对管道、FIFO 和记录锁的讨论侧重于从它们的 Posix 定义出发。
- 本卷不仅描述了 IPC 机制及其使用方法，还实现了 Posix 消息队列、读写锁与 Posix 信号量（都可以实现为用户库）。这些实现可以把多种不同的特性捆绑起来（例如，Posix 信号量的一种实现用到了互斥量、条件变量和内存映射 I/O），还强

调了我们在应用程序中经常要处理的一些问题（如竞争状态、错误处理、内存泄漏和变长参数列表）。理解某种特性的实现通常有助于了解如何使用该特性。

- 对 RPC 的讨论侧重于 Sun 的 RPC 包。在此之前讲述了新的 Solaris 门 API，它类似于 RPC 但用于单主机。这么一来我们就介绍了许多在调用其他进程中的过程时需要考虑的特性，而不用关心网络方面的细节。

读者对象

本书既可以用作 IPC 的教程，也可以用作有经验的程序员的参考书。全书划分为以下 4 个主要部分：

- 消息传递；
- 同步；
- 共享内存；
- 远程过程调用。

但许多读者可能只对特定的部分感兴趣。第 2 章总结了所有 Posix IPC 函数共有的特性，第 3 章归纳了所有 System V IPC 函数共有的特性，第 12 章介绍了 Posix 和 System V 的共享内存，但书中多数章节都可以独立于其他章节阅读。所有读者都应该阅读第 1 章，尤其是 1.6 节，该节介绍了一些贯穿全书的包装函数。讨论 Posix IPC 的各章与讨论 System V IPC 的各章彼此独立，有关管道、FIFO 和记录锁的几章不属于上述两个阵营，关于 RPC 的两章也独立于其他 IPC 方法。

为了方便读者把本书作为参考书，本书提供了完整的全文索引，并在最后几页总结了每个函数和结构的详细描述在正文中的哪里可以找到。为了给不按顺序阅读本书的读者提供方便，我们在书中为各个主题提供了大量的交叉引用。

源代码与勘误

书中所有示例的源代码可以从作者主页（列在前言的最后）获得[①]。学习本书讲述的 IPC 技术的最好方法就是下载这些程序，对其进行修改和改进。只有这样实际编写代码才能深入理解有关概念和方法。每章末尾提供了大量的习题，大部分在附录 D 中给出答案。

本书的最新勘误表也可以从作者主页获取。

致谢

尽管封面上只出现了作者一个人的名字，但一本高质量的书的出版需要许多人的共同努力。首先要感谢我的家人，他们在我写书的那段时间里承担了一切。再次感谢你

① 书中所有示例的源代码也可以从异步社区http: //www.epubit.com.cn/book/details/4088免费注册下载。——编者注

们：Sally、Bill、Ellen 和 David。

感谢技术审稿人给出的宝贵的反馈意见（打印出来有 135 页）。他们发现了许多错误，指出了需要更多解释的地方，并对表达、用词和代码提出了许多修改建议，他们是 Gavin Bowe、Allen Briggs、Dave Butenhof、Wan-Teh Chang、Chris Cleeland、Bob Friesenhahn、Andrew Gierth、Scott Johnson、Marty Leisner、Larry McVoy、Craig Metz、Bob Nelson、Steve Rago、Jim Reid、Swamy K. Sitarama、Jon C. Snader、Ian Lance Taylor、Rich Teer 和 Andy Tucker。

下列诸位通过电子邮件回答过我的问题，有人甚至回答过很多问题。澄清这些问题提高了本书的准确性并改进了语言表达，他们是 David Bausum、Dave Butenhof、Bill Gallmeister、Mukesh Kacker、Brian Kernighan、Larry McVoy、Steve Rago、Keith Skowran、Bart Smaalders、Andy Tucker 和 John Wait。

特别感谢GSquared的Larry Rafsky提供了很多帮助。像以往一样，感谢国家光学天文台（NOAO）、Sidney Wolff、Richard Wolff和Steve Grandi，他们为我提供了网络与主机的访问权限。DEC公司的Jim Bound、Matt Thomas、Mary Clouter和Barb Glover提供了用于本书多数示例的Alpha系统。书中的一部分代码是在其他Unix系统上测试的：感谢Red Hat软件公司的Michael Johnson提供了最新版本的Red Hat Linux，感谢IBM奥斯汀实验室的Dave Marquardt和Jessie Haug提供了RS/6000系统以及最新版本的AIX的访问权限。

最后还要感谢 Prentice Hall 的优秀员工（本书的编辑 Mary Franz，还有 Noreen Regina、Sophie Papanikolaou 和 Patti Guerrieri）给予的帮助，尤其是在很紧的时间内完成一切所付出的努力。

版权说明

我制作了本书的最终电子版（PostScript 格式），最后排版成现在的书。我用 James Clark 编写的优秀的 `groff` 包为本书排版，该软件包安装在一台运行 Solaris 2.6 的 SparcStation 工作站上。（认为 troff 已经过时的报导显然太夸张了。）我使用 `vi` 编辑器键入了所有的 138 897 个单词，用 `gpic` 程序绘制了 72 幅插图（其中用到了许多由 Gary Wright 编写的宏），用 `gtbl` 程序生成了 35 张表格，为全书添加了索引（用到了 Jon Bentley 与 Brian Kernighan 编写的一组 `awk` 脚本），并设计了最终的版式。我录入书中的 8 046 行 C 语言源代码，使用的是 Dave Hanson 的 `loom` 程序、GNU 的 `indent` 程序和 Gary Wright 写的一些脚本。

欢迎读者以电子邮件的方式反馈意见、提出建议或订正错误。

W. Richard Stevens
1998 年 7 月于亚利桑那州图森市
http://www.kohala.com/~rstevens

目　录

Function prototype	page
`bool_t clnt_control(CLIENT *cl, unsigned int request, char *ptr);`	418
`CLIENT *clnt_create(const char *host, unsigned long prognum, unsigned long versnum, const char *protocol);`	401
`void clnt_destroy(CLIENT *cl);`	420
`int door_bind(int fd);`	390
`int door_call(int fd, door_arg_t *argp);`	361
`int door_create(Door_server_proc *proc, void *cookie, u_int attr);`	363
`int door_cred(door_cred_t *cred);`	365
`int door_info(int fd, door_info_t *info);`	365
`int door_return(char *dataptr, size_t datasize, door_desc_t *descptr, size_t ndesc);`	365
`int door_revoke(int fd);`	390
`Door_create_proc *door_server_create(Door_create_proc *proc);`	384
`int door_unbind(void);`	390
`void err_dump(const char *fmt, ...);`	512
`void err_msg(const char *fmt, ...);`	512
`void err_quit(const char *fmt, ...);`	512
`void err_ret(const char *fmt, ...);`	511
`void err_sys(const char *fmt, ...);`	511
`int fcntl(int fd, int cmd, ... /* struct flock *arg */ );`	199
`int fstat(int fd, struct stat *buf);`	328
`key_t ftok(const char *pathname, int id);`	28
`int ftruncate(int fd, off_t length);`	327
`int mkfifo(const char *pathname, mode_t mode);`	54
`void *mmap(void *addr, size_t len, int prot, int flags, int fd, off_t offset);`	308
`int msync(void *addr, size_t len, int flags);`	310
`int munmap(void *addr, size_t len);`	309
`int mq_close(mqd_t mqdes);`	77
`int mq_getattr(mqd_t mqdes, struct mq_attr *attr);`	79
`int mq_notify(mqd_t mqdes, const struct sigevent *notification);`	87
`mqd_t mq_open(const char *name, int oflag, ... /* mode_t mode, struct mq_attr *attr */ );`	76
`ssize_t mq_receive(mqd_t mqdes, char *ptr, size_t len, unsigned int *priop);`	83
`int mq_send(mqd_t mqdes, const char *ptr, size_t len, unsigned int prio);`	83
`int mq_setattr(mqd_t mqdes, const struct mq_attr *attr, struct mq_attr *oattr);`	79
`int mq_unlink(const char *name);`	77
`int msgctl(int msqid, int cmd, struct msqid_ds *buff);`	134
`int msgget(key_t key, int oflag);`	130
`ssize_t msgrcv(int msqid, void *ptr, size_t length, long type, int flag);`	132
`int msgsnd(int msqid, const void *ptr, size_t length, int flag);`	131
`int pclose(FILE *stream);`	52
`int pipe(int fd[2]);`	44
`FILE *popen(const char *command, const char *type);`	52

Function prototype	page
`int pthread_cancel(pthread_t tid);`	187
`void pthread_cleanup_pop(int execute);`	187
`void pthread_cleanup_push(void (*function)(void *), void *arg);`	187
`int pthread_create(pthread_t *tid, const pthread_attr_t *attr, void *(*func)(void *), void *arg);`	502
`int pthread_detach(pthread_t tid);`	504
`void pthread_exit(void *status);`	504
`int pthread_join(pthread_t tid, void **status);`	503
`pthread_t pthread_self(void);`	503
`int pthread_condattr_destroy(pthread_condattr_t *attr);`	172
`int pthread_condattr_getpshared(const pthread_condattr_t *attr, int *valptr);`	173
`int pthread_condattr_init(pthread_condattr_t *attr);`	172
`int pthread_condattr_setpshared(pthread_condattr_t *attr, int value);`	173
`int pthread_cond_broadcast(pthread_cond_t *cptr);`	171
`int pthread_cond_destroy(pthread_cond_t *cptr);`	172
`int pthread_cond_init(pthread_cond_t *cptr, const pthread_condattr_t *attr);`	172
`int pthread_cond_signal(pthread_cond_t *cptr);`	167
`int pthread_cond_timedwait(pthread_cond_t *cptr, pthread_mutex_t *mptr, const struct timespec *abstime);`	171
`int pthread_cond_wait(pthread_cond_t *cptr, pthread_mutex_t *mptr);`	167
`int pthread_mutexattr_destroy(pthread_mutexattr_t *attr);`	172
`int pthread_mutexattr_getpshared(const pthread_mutexattr_t *attr, int *valptr);`	173
`int pthread_mutexattr_init(pthread_mutexattr_t *attr);`	172
`int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr, int value);`	173
`int pthread_mutex_destroy(pthread_mutex_t *mptr);`	172
`int pthread_mutex_init(pthread_mutex_t *mptr, const pthread_mutexattr_t *attr);`	172
`int pthread_mutex_lock(pthread_mutex_t *mptr);`	160
`int pthread_mutex_trylock(pthread_mutex_t *mptr);`	160
`int pthread_mutex_unlock(pthread_mutex_t *mptr);`	160
`int pthread_rwlockattr_destroy(pthread_rwlockattr_t *attr);`	179
`int pthread_rwlockattr_getpshared(const pthread_rwlockattr_t *attr, int *valptr);`	179
`int pthread_rwlockattr_init(pthread_rwlockattr_t *attr);`	179
`int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *attr, int value);`	179
`int pthread_rwlock_destroy(pthread_rwlock_t *rwptr);`	179
`int pthread_rwlock_init(pthread_rwlock_t *rwptr, const pthread_rwlockattr_t *attr);`	179
`int pthread_rwlock_rdlock(pthread_rwlock_t *rwptr);`	178
`int pthread_rwlock_tryrdlock(pthread_rwlock_t *rwptr);`	178
`int pthread_rwlock_trywrlock(pthread_rwlock_t *rwptr);`	178
`int pthread_rwlock_unlock(pthread_rwlock_t *rwptr);`	178
`int pthread_rwlock_wrlock(pthread_rwlock_t *rwptr);`	178

Part 1

Introduction
简介