Addison Wesley

TURING

图灵原版计算机科学系列

# UNIX 网络编程

## 卷2：进程间通信

英文版·第2版

# UNIX Network Programming

## Volume 2: Interprocess Communications, *Second Edition*

[美] W. Richard Stevens 著

人民邮电出版社
POSTS & TELECOM PRESS

TURING

图灵原版计算机科学系列

# UNIX 网络编程

## 卷2：进程间通信

英文版·第2版

# UNIX Network Programming

Volume 2: Interprocess Communications, *Second Edition*

[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＝UNIX Network Programming:第2版．第2卷．进程间通信：英文/（美）史蒂文斯（Stevens,W.R.）著．—北京：人民邮电出版社，2009.11

（图灵原版计算机科学系列）

ISBN 978-7-115-21511-6

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（2009）第172154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。

图灵原版计算机科学系列

**UNIX网络编程　卷2：进程间通信（英文版·第2版）**

◆ 著　　[美] W. Richard Stevens

责任编辑　杨海玲

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号

邮编　100061　　电子函件　315@ptpress.com.cn

网址　http://www.ptpress.com.cn

北京隆昌伟业印刷有限公司印刷

◆ 开本：800×1000　1/16

印张：36

字数：692千字　　2009年11月第1版

印数：1-2 500册　　2009年11月北京第1次印刷

著作权合同登记号　图字：01-2009-5714号

ISBN 978-7-115-21511-6

定价：89.00元

**读者服务热线：（010）51095186　印装质量热线：（010）67129223**

**反盗版热线：（010）67171154**

# 版 权 声 明

# Preface
# 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、

①书中所有示例的源代码也可以从图灵网站（www.turingbook.com）本书网页免费注册下载。——编者注

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

# Contents

# 目　录

# Part 1

# Introduction
# 简介

# 1

# Introduction
# 简介

## 1.1 Introduction 概述

IPC stands for *interprocess communication*. Traditionally the term describes different ways of *message passing* between different processes that are running on some operating system. This text also describes numerous forms of *synchronization*, because newer forms of communication, such as shared memory, require some form of synchronization to operate.

In the evolution of the Unix operating system over the past 30 years, message passing has evolved through the following stages:

- *Pipes* (Chapter 4) were the first widely used form of IPC, available both within programs and from the shell. The problem with pipes is that they are usable only between processes that have a common ancestor (i.e., a parent–child relationship), but this was fixed with the introduction of *named pipes* or *FIFOs* (Chapter 4).
- *System V message queues* (Chapter 6) were added to System V kernels in the early 1980s. These can be used between related or unrelated processes on a given host. Although these are still referred to with the "System V" prefix, most versions of Unix today support them, regardless of whether their heritage is System V or not.

When describing Unix processes, the term *related* means the processes have some ancestor in common. This is another way of saying that these related processes were generated