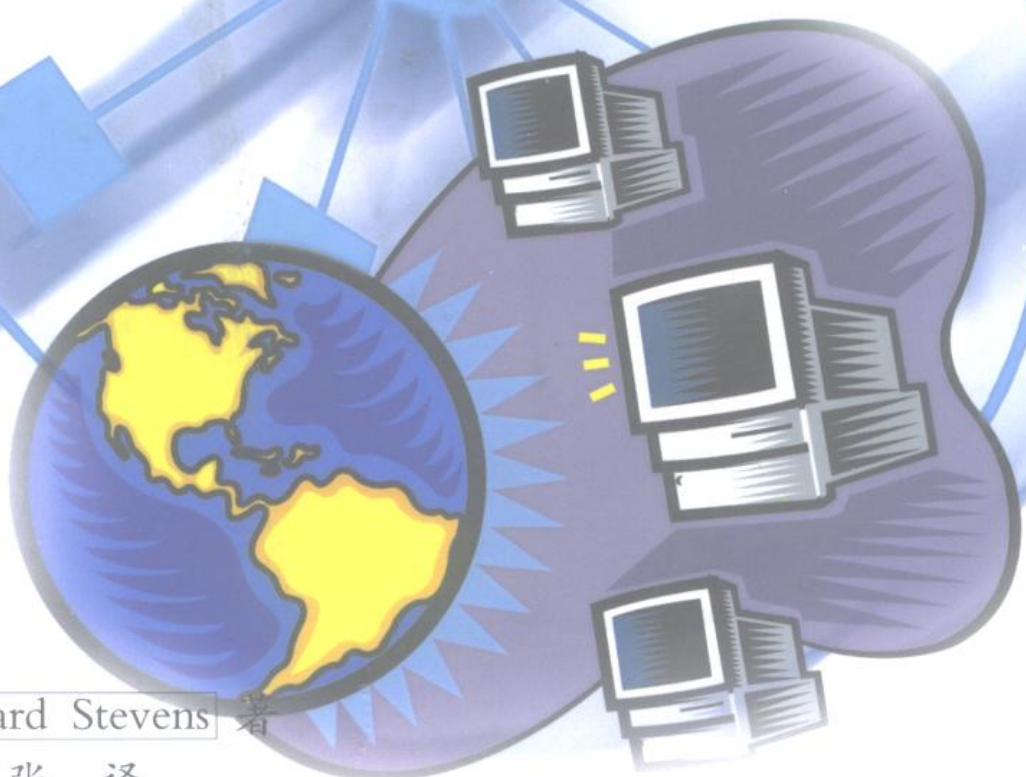
北京科海培训中心

Prentice Hall PTR

UNIX 网络编程

第2卷：进程间通信

第2版



[美] W. Richard Stevens 著

杨继张 译

清华大学出版社

Prentice Hall PTR　　北京科海培训中心

UNIX 网络编程(第 2 版)

第 2 卷:进程间通信

[美] W. Richard Stevens 著

杨继张 译

清 华 大 学 出 版 社

(京)新登字 158 号

著作权合同登记号:01-1999-3354

内 容 提 要

本书全面深入地讲述了各种进程间通信(IPC)形式,它们是几乎所有复杂精致的 UNIX 程序的性能之关键。从网络编程角度看,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 的内部结构的综合讨论开始,具体阐述并比较了四种 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、信号灯)、共享内存区(匿名共享内存区、有名共享内存区)及远程过程调用(Solaris 门、Sun RPC)。在附录中给出了测量各种 IPC 形式之性能的方法。

本书内容详尽且具权威性,几乎每章都提供精选的习题,是计算机和网络专业高年级本科生和研究生的首选教材。本书也可作为网络研究和开发人员的自学教材和参考书。

书　名:UNIX 网络编程(第 2 版)第 2 卷:进程间通信

作　者:W. Richard Stevens

译　者:杨继张

出版者:清华大学出版社(北京清华大学校内,邮编 100084)

印刷者:北京门头沟胶印厂

发　行:新华书店总店北京科技发行所

开　本:16　印张:30.75　字数:748 千字

版　次:2000 年 3 月第 1 版　　2000 年 7 月第 3 次印刷

印　数:8001~10000

书　号:ISBN 7-302-03815-5/TP・2232

定　价:58.00 元

谨以此中译本奉献

本书原作者 W. Richard Stevens 先生在天之英灵。

——Stevens 先生是学界知名的网络与 Unix 专家、教育家，

不幸已于 1999 年 9 月 1 日溘然长逝，终年 48 岁。

译者序

译者直到去年9月中旬才惊悉本书作者W. Richard Stevens博士已逝世的噩耗。在此之前译者只是奇怪有相当长的一段时间访问不了Stevens博士的个人主页，并没想到其背后隐蔽着这件令全世界计算机界和网络工作者为之扼腕叹息的事。当时本书的翻译工作尚未开始，译者也曾一度在是否接手翻译上徘徊，因为当时本人恰好有一个作为访问学者赴美工作的机会，然而自从参与翻译与审校本丛书第1卷以来，译者就深深地为Stevens博士的博学和极为严谨的治学风范所折服，由此产生了将整套丛书翻译成中文的使命感，第1卷中译本的广大读者以电子邮件方式向译者传递的赞誉、鼓励和希望尽早看到后续各卷中译本的心情也在敦促译者继续努力。Stevens博士与世长辞的消息最终令译者下定决心翻译出本卷——Stevens博士最后的著作。在让这部遗著的中译本尽快面世的内在动力推动下，经过近两个月夜以继日的工作，中译本初稿终于完成。

本书是《UNIX网络编程(第2版)》(UNIX Network Programming)这套计划中的三卷本丛书中第2卷的中译本。本丛书第1卷的副标题为“套接口API和X/Open传输接口API”(Networking APIs: Sockets and XTI)，讨论的是让连网的不同主机彼此交换信息的应用程序的编写细节。第2卷的副标题为“进程间通信”(Interprocess Communications)，讨论的是同一主机内不同进程或线程彼此交换信息的应用程序的编写细节。按照Stevens博士原来的计划，第3卷的副标题为“应用程序”(Applications)，它是对《UNIX网络编程(第1版)》(1990年)中第9至第18章内容的扩充，可想其内容之丰富。永远遗憾的是Stevens博士已不可能完成第3卷了，不过像Gary R. Wright等人也许能够把它整理并续写出来。Wright是Stevens博士另一套丛书即《TCP/IP阐述》(TCP/IP Illustrated)中第2卷的合作者，Stevens博士的个人主页也是他于去年10月初重新开通的。

本书作者Stevens博士的生平在《UNIX网络编程(第2版)》第1卷中译本的译者序中已经提到过。Stevens博士尽管时间非常宝贵，每天还要花不少时间阅读和回答读者们发给他的有关Unix和TCP/IP的电子邮件，因而颇受尊敬。Stevens博士本人也在与网友们的交往中获益不少，他在本书扉页上写的话就是：“献给Usenet社群；谢谢他们回答了许多问题，又提供了不少FAQ。”

与《UNIX网络编程(第2版)》第1卷的翻译一样，译者始终以教科书的要求认真对待本卷的翻译与审校工作。除根据作者给出的勘误表进行修正外，译者还就若干不易理解或易混淆的概念和说法给出了自己的见解和补充说明。本卷的翻译继承了第1卷中译本中的大多数说法，特别是其中以译者注形式强调的概念，本书不再重复说明。尽管如此，由于译者水平有限，译文中仍难免有不妥之处，敬请广大读者批评指正。就中译本中的技术问题，读者可直接与译者本人联系，电子邮件地址为jzyang@chpcc.edu.cn，通信地址为清华大学网络中心。

译者最后特别感谢北京科海培训中心编辑室的全体同志，本书的出版与他们默默无闻的辛勤工作和热情的支持是分不开的。

译 者

2000 年 1 月

前　言

简介

多数精致复杂的程序涉及某种形式的 IPC，也就是进程间通信（Interprocess Communication）。它是一个程序设计原则的自然结果，即把应用程序设计成一组彼此通信的小片段是比设计成单个庞大的程序更好的方法。从历史上看，应用程序设计方式按如下的顺序渐次出现：

1. 完成全部工作的单个庞大的程序。整个程序的各种片段可作为函数实现，它们以函数参数、函数返回值及全局变量的形式彼此交换信息。
2. 使用某种形式的 IPC 彼此通信的多个程序。许多标准 Unix 工具就是以这种样式设计的，它们使用 shell 管道（一种 IPC 形式）从一个程序向下一个程序传递信息。
3. 由使用某种形式的 IPC 彼此通信的多个程序构成的单个程序。尽管这种通信发生在线程间而不是进程间，我们仍用 IPC 的说法来描述。

把后两种设计方式结合起来也是可能的：由多个进程组成，每个进程又由一个或多个线程构成，其中涉及给定进程内各线程间的通信以及不同进程间的通信。

到此为止所描述的是把完成一个给定应用所涉及的工作散布到多个进程中，也许还散布到进程内的线程中。在含有多个处理器（CPU）的系统上，多个进程可能（在不同的 CPU 上）同时运行，一个给定进程的多个线程也可能同时运行。因此，把一个应用的工作散布到多个进程或线程中有可能减少该应用完成给定任务的时间。

本书具体叙述 4 种不同形式的 IPC：

1. 消息传递（管道、FIFO、消息队列）
2. 同步（互斥锁、条件变量、读写锁、文件与记录锁、信号灯）
3. 共享内存区（匿名共享内存区、有名共享内存区）
4. 远程过程调用（Solaris 门、Sun RPC）

本书不讨论通过计算机网络通信的程序的编写。这种通信形式通常涉及使用 TCP/IP 协议族的所谓的套接口 API（应用程序编程接口）；这些主题在本丛书的第 1 卷[Stevens 1998]中详细讨论。

有人可能坚称单台主机内的即不涉及网络的 IPC（正是本卷的主题）不应该使用，相反，所有应用程序都应该编写成通过网络运行在各种主机的分布式应用程序。然而实践证明，单台主机内的 IPC 与穿越网络的通信相比，前者往往快得多，有时还简单些。诸如共享内存区和同步这样的技术通常只在单台主机上可用，穿越网络时可能没法用。经验与历史告诉我们，不涉及网络的 IPC（本卷）和穿越网络的 IPC（本丛书第 1 卷）都有用武之地。

本书构筑在同一套丛书第 1 卷以及作者的其他 4 本书的基础之上，它们的书名在全书中缩写如下：

- UNPv1:*UNIX Network Programming*, *Volume* 1 [Stevens 1998]①
- 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]

尽管在以"网络编程"为书名的丛书中讨论 IPC 看起来可能奇怪,IPC 却往往用在网络应用程序中。这正如在"*UNIX Network Programming*" 1990 年版的前言中所说:"理解如何给一个网络开发软件的必要条件之一是理解进程间通信(IPC)"。

本书与第 1 版的差别

本卷是对"*UNIX Network Programming*"1990 年版中第 3 章和第 18 章的完全重写和扩充。从词数上统计,整个材料扩充了 5 倍。下面是新版本中所做的主要修改:

- 除了"System V IPC"的 3 种形式(消息队列、信号灯、共享内存区)外,还讨论了实现这 3 种形式 IPC 的更新的 Posix 函数。(我们将在 1.7 节谈一谈 Posix 标准族。)作者期待以后数年内 Posix IPC 函数有大的推广,它们与对等的 System V 函数相比毕竟有优势。
- 讨论了 Posix 用于同步的函数:互斥锁、条件变量、读写锁。这些函数既可用于同步进程,也可用于同步线程,而且往往在访问共享内存区时使用。
- 本卷假设有一个 Posix 线程环境(称为"Pthreads"),许多例子设计成使用多个线程而不是多个进程。
- 关于管道、FIFO 和记录上锁的讨论集中于它们的 Posix 定义。
- 除讲述 IPC 机制并展示如何使用它们外,作者还开发了 Posix 消息队列、读写锁及 Posix 信号灯的实现(所有这些 IPC 机制都可作为用户函数库实现)。这些实现能够把许多不同的特性联结在一起(例如有一个 Posix 信号灯的实现同时用上了互斥锁、条件变量和内存映射 I/O),并强调了在我们的应用程序中必须经常处理的条件(例如竞争状态、出错处理、内存空间遗漏、可变长度参数表等)。理解某个特性的一种实现往往导致在如何使用该特性上取得更大的认识。
- 关于 RPC 的讨论集中于 Sun RPC 软件包。在此之前我们将讲述新的 Solaris 门 API,它与 RPC 类似,但限于单台主机。这么一来介绍了许多新特性,它们是调用另一个进程中的过程时需要关心的,不过网络连接的具体细节不必担心。

读者

本书既可作为关于 IPC 的指导书,也可作为有经验程序员的参考书。它分为 4 个部分:

- 消息传递
- 同步
- 共享内存区

---

① 译者注: 本书有中译本(清华大学出版社出版),在出现指向本书具体页码的参考点处,我们将注出中译本中的对应页码。

· 远程过程调用

不过许多读者可能只对其中的特定子集感兴趣。多数单章能彼此独立地分开阅读，但是第 2 章汇总了所有的 Posix IPC 函数共同的许多特性，第 3 章汇总了所有的 System V IPC 函数共同的许多特性，第 12 章则是对 Posix 和 System V 共享内存区的笼统介绍。所有读者都应阅读第 1 章，特别是 1.6 节，它们介绍了全书都用到的一些包裹函数。讨论 Posix IPC 的各章与讨论 System V IPC 的各章彼此独立，关于管道、FIFO 和记录上锁的各章则不属于任何一个阵营。关于 RPC 的两章也不同于其他 IPC 技术。

为便于用作参考书，本书提供了全文索引，并在附录 F 和附录 G 中给出了所有的函数和结构的具体讲解所在的页码。为帮助那些以随意顺序阅读各主题的读者，全文提供了大量的对相关主题的参考点。

源代码和勘误表获取

本书中出现的所有例子的源代码都可从作者的主页获取，其 URL 地址列在本前言的末尾。学习本书中讲述的 IPC 技术的最好方法是使用这些程序，然后修改并改进它们。只有真正编写这种形式的代码，才能加深对概念的理解并提高编程技巧。各章最后提供了大量的习题，附录 D 给出了其中大多数的解答。

本书最新的勘误表也可从作者的主页获取②。

鸣谢

尽管作者的名字是唯一出现在封面上的，要创作一本高质量的教科书却免不了许多人的共同努力。首先应该而且最值得感谢的是作者的家庭成员，他们几乎天天承受着作者写书时的那些个漫长而难熬的钟点。再次谢谢你们，Sally、Bill、Ellen、David。

作者感谢本书的技术性评阅人，他们提供了无价的反馈信息（打印出来共 135 页），帮助作者找出许多错误，指出需要进一步解释的地方，并提出另外的表述、用词和编码方案，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。

下面列出的网友回答了作者的电子邮件中的问题——有些网友还回答了不少，所有这些答复都有助于改进本书的精确性和语言表达，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，原因很多。跟以往一样，作者感谢国家光学天文台（NOAO）、Sidney Wolff、Richard Wolff 和 Steve Grandi，他们允许作者访问他们的网络和主机。DEC 公司的 Jim Bound、Matt Thomas、Mary Clouter 和 Barb Glover 提供的 Alpha 系统正是运行过本书中大多数例子的那台主机。本书中代码的一个子集是在其他 Unix 系统上测试的：作者感谢 Red Hat 软件公司的 Michael Johnson 提供了 Red Hat Linux 的最新版本，

② 译者注：中译本已根据最后修改日期为 1999 年 8 月 27 的最新勘误表作过订正。

IBM Austin 的 Dave Marquardt 和 Jessie Hang 提供了一台 RS/6000 系统以及最新版本的 AIX。

作者感谢 Prentice Hall 出版社那些勤勤恳恳的职员，谢谢他们的所有帮助，特别是在时间紧的情况下还能井然有序地把事情做好，他们是编辑 Mary Franz 以及编务 Noreen Regina、Sophie Papanikolaou 和 Patti Guerrieri。

附记

作者制作了本书的可照排拷贝(PostScript 格式)，这个拷贝随后排版成成品书。所用的格式化系统是 James Clark 的优秀的 groff 软件包，它安装在运行 Solaris 2.6 的一台 SparcStation 工作站上。(宣判 troff 死刑的报告毫无疑问过度夸张了。)作者使用 vi 编辑器键入了总共138 897个单词，使用 gpic 程序制作了 72 张插图(用到了 Gary Wright 编写的许多宏)，使用 gtbl 程序生成了 35 个表格，执行了所有的索引任务(使用由 John Bentley 和 Brian Kernighan 编写的一组 awk 脚本)，并完成了最终的页面布局。Dave Hanson 的 loom 程序、GNU indent 程序以及由 Gary Wright 编写的一些脚本程序用于把总共 8046 行 C 源代码插入到本书中。

作者欢迎任何读者发电子邮件提出意见或建议，或者给出程序缺陷的补丁。

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

目 录

第1部分 简 介

第2部分 消息传递

第3部分 同 步

第 6 部分 附 录

第1部分 简 介

第1章 简 介

1.1 概 述

IPC是进程间通信(interprocess communication)的简称。传统上该术语描述的是运行在某个操作系统之上的不同进程间消息传递(message passing)的不同方式。本书还讲述多种形式的同步(synchronization),因为像共享内存区这样的较新式的通信需要某种形式的同步参与运作。

在Unix操作系统过去30年的演变史中,消息传递历经了如下几个发展阶段:

- 管道(pipe,第4章)是第一个广泛使用的IPC形式,既可在程序中使用,也可从shell中使用。管道的问题在于它们只能在具有共同祖先(指父子进程关系)的进程间使用,不过该问题已随有名管道(named pipe)即FIFO(第4章)的引入而解决了。
- System V消息队列(System V message queue,第6章)是在20世纪80年代早期加到System V内核中的。它们可用在同一主机上有亲缘关系或无亲缘关系的进程之间。尽管称呼它们时仍冠以"System V"前缀,当今多数版式的Unix却不论自己是否源自System V都支持它们。

  在谈论Unix进程时,有亲缘关系(related)的说法意味着所论及的进程具有某个共同的祖先。说得更明白点,这些有亲缘关系的进程是从该祖先进程经过一次或多次fork派生来的。一个常见的例子是在某个进程调用fork两次,派生出两个子进程之后。我们说这两个子进程是有亲缘关系的。同样,每个子进程与其父进程也是有亲缘关系的。考虑到IPC,父进程可以在调用fork前建立某种形式的IPC(例如管道或消息队列),因为它知道随后派生的两个子进程将穿越fork继承该IPC对象。我们随图1.6详细讨论各种IPC对象的继承性。

  我们还得注意,从理论上说,所有Unix进程与init进程都有亲缘关系,它是在系统自举时启动所有初始化进程的祖先进程。然而从实践上说,进程亲缘关系开始于一个登录shell(称为一个会话)以及由该shell派生的所有进程。APUE的第9章详细讨论会话和进程亲缘关系。

  本书将全文使用缩进的插入式注解(如此处所示的版本)来说明实现上的细节、历史上的观点以及其他琐事。

- Posix消息队列(Posix消息队列,第5章)是由Posix实时标准(1003.1b-1993,将在1.7节详细讨论)加入的。它们可用在同一主机上有亲缘关系和无亲缘关系的进程