Addison Wesley

TURING

图灵计算机科学丛书

UNIX 网络编程

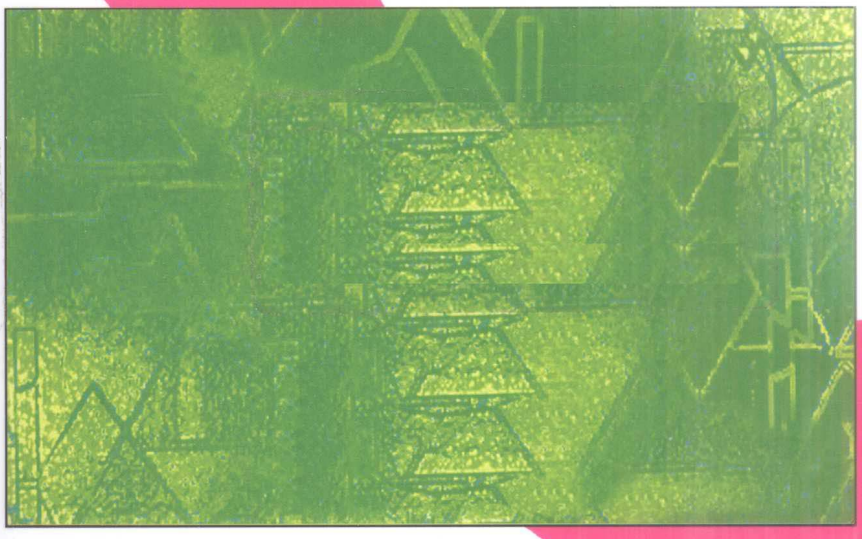
卷2：进程间通信

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

UNIX 网络编程

卷2：进程间通信

第2版

UNIX Network Programming

[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版. 第2卷, 进程间通信 / (美)史蒂文斯 (Stevens, W. R.) 著. — 北京 : 人民邮电出版社, 2010.7
(图灵计算机科学丛书)
书名原文: UNIX Network Programming, Volume 2: Interprocess Communications
ISBN 978-7-115-23028-7

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

中国版本图书馆CIP数据核字(2010)第086883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。

图灵计算机科学丛书

UNIX网络编程 卷2：进程间通信（第2版）

◆ 著 [美] W. Richard Stevens
责任编辑 杨海玲
执行编辑 傅尔也

◆ 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
三河市海波印务有限公司印刷

◆ 开本：787×1092 1/16
印张：29.5
字数：830千字 2010年7月第1版
印数：1－4 000册 2010年7月河北第1次印刷

著作权合同登记号 图字：01-2009-5716号

ISBN 978-7-115-23028-7

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

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、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

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

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

第一部分　简介

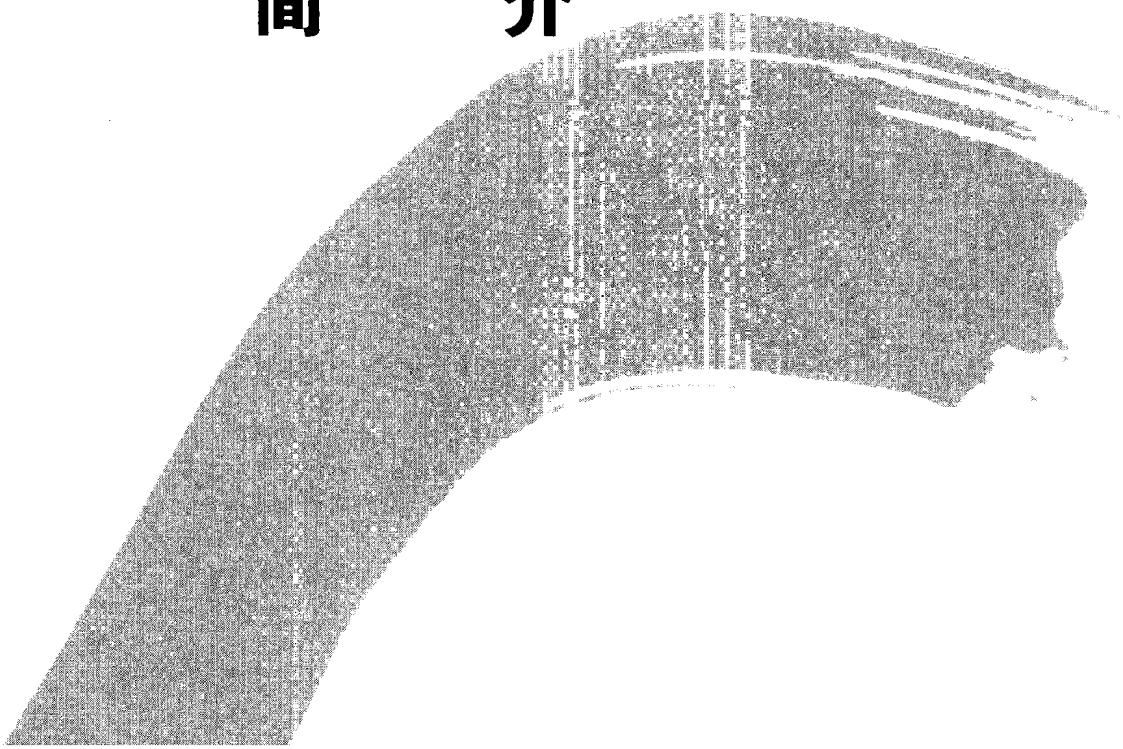
第二部分　消息传递

第三部分 同步

第四部分 共享内存区

第一部分

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各种消息传递（message passing）的方式。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，当今多数版本的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。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）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之间。
- 远程过程调用（Remote Procedure Call，简称RPC，第5部分）出现在20世纪80年代中期，它是从一个系统（客户主机）上某个程序调用另一个系统（服务器主机）上某个函数的一种方法，是作为显式网络编程的一种替换方法开发的。既然客户和服务器之间通常传递一些信息（被调用函数的参数与返回值），而且RPC可用在同一主机上的客户和服务器之间，因此可认为RPC是另一种形式的消息传递。

看一看由Unix提供的各种同步形式的演变同样颇有教益。

- 需要某种同步形式（往往是为了防止多个进程同时修改同一文件）的早期程序使用了文件系统的诡秘特性，我们将在9.8节讨论其中的一些。
- 记录上锁（record locking，第9章）是在20世纪80年代早期加到Unix内核中的，然后在1988年由Posix.1标准化的。
- System V信号量（System V semaphore，第11章）是在System V消息队列加入System V内核的同时（20世纪80年代早期）伴随System V共享内存区（System V shared memory）加入的。当今多数版本的Unix都支持它们。
- Posix信号量（Posix semaphore，第10章）和Posix共享内存区（Posix shared memory，第13章）也由Posix实时标准（1003.1b-1993）加入。
- 互斥锁（mutex）和条件变量（condition variable，见第7章）是由Posix线程标准（1003.1c-1995）定义的两种同步形式。尽管往往用于线程间的同步，它们也能提供不同进程间的同步。
- 读写锁（read-write lock，第8章）是另一种形式的同步。它们还没有被Posix标准化，不过也许不久后会被标准化。

1.2 进程、线程与信息共享

按照传统的Unix编程模型，我们在一个系统上运行多个进程，每个进程都有各自的地址空间。Unix进程间的信息共享可以有多种方式。图1-1对此作了总结。

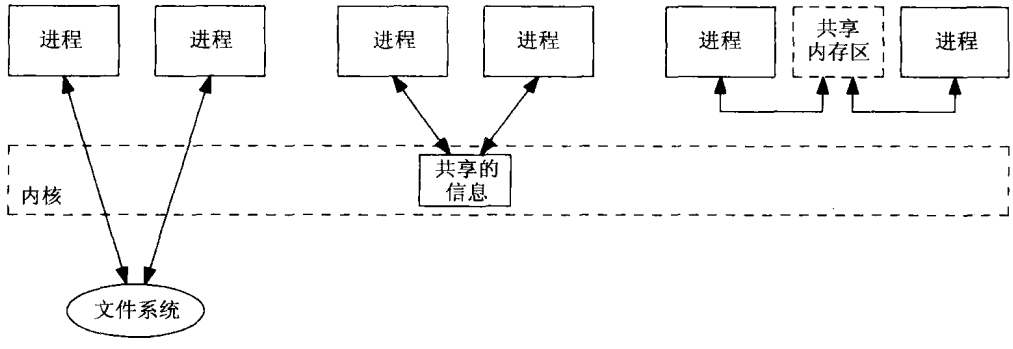


图1-1 Unix进程间共享信息的三种方式

(1) 左边的两个进程共享存留于文件系统中某个文件上的某些信息。为访问这些信息，每个进程都得穿越内核（例如read、write、lseek等）。当一个文件有待更新时，某种形式的同步是必要的，这样既可保护多个写入者，防止相互串扰，也可保护一个或多个读出者，防止写入者的干扰。

(2) 中间的两个进程共享驻留于内核中的某些信息。管道是这种共享类型的一个例子，System V消息队列和System V信号量也是。现在访问共享信息的每次操作涉及对内核的一次系统调用。

(3) 右边的两个进程有一个双方都能访问的共享内存区。每个进程一旦设置好该共享内存区，就能根本不涉及内核而访问其中的数据。共享该内存区的进程需要某种形式的同步。

注意没有任何东西限制任何IPC技术只能使用两个进程。我们讲述的技术适用于任意数目的进程。在图1-1中只展示两个进程是为了简单起见。

线程

5 虽然Unix系统中进程的概念已使用了很久，一个给定进程内多个线程（thread）的概念却相对较新。Posix.1线程标准（称为“Pthreads”）是于1995年通过的。从IPC角度看来，一个给定进程内的所有线程共享同样的全局变量（也就是说共享内存区的概念对这种模型来说是内在的）。然而我们必须关注的是各个线程间对全局数据的同步访问。同步尽管不是一种明确的IPC形式，但它确实伴随许多形式的IPC使用，以控制对某些共享数据的访问。

本书中我们讲述进程间的IPC和线程间的IPC。我们假设有一个线程环境，并作类似如下形式的陈述：“如果管道为空，调用线程就阻塞在它的read调用上，直到某个线程往该管道写入数据。”要是你的系统不支持线程，那你可以将该句子中的“线程”替换成“进程”，从而提供“阻塞在对空管道的read调用上”的经典Unix定义。然而在支持线程的系统上，只有对空管道调用read的那个线程阻塞，同一进程中的其余线程才可以继续执行。向该空管道写数据的工作既可以由同一进程中的另一个线程去做，也可以由另一个进程中的某个线程去做。

附录B汇总了线程的某些特征以及全书都用到的5个基本的Pthread函数。

1.3　IPC对象的持续性

我们可以把任意类型的IPC的持续性（persistence）定义成该类型的一个对象一直存在多长时间。图1-2展示了三种类型的持续性。

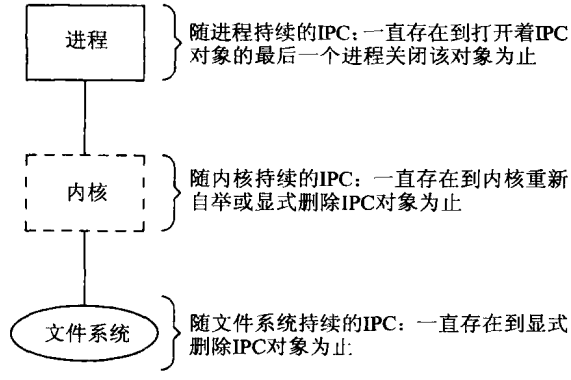


图1-2　IPC对象的持续性

(1) 随进程持续的（process-persistent）IPC对象一直存在到打开着该对象的最后一个进程关闭该对象为止。例如管道和FIFO就是这种对象。

(2) 随内核持续的（kernel-persistent）IPC对象一直存在到内核重新自举或显式删除该对象为止。例如System V的消息队列、信号量和共享内存区就是此类对象。Posix的消息队列、信号
6 量和共享内存区必须至少是随内核持续的，但也可以是随文件系统持续的，具体取决于实现。

(3) 随文件系统持续的（filesystem-persistent）IPC对象一直存在到显式删除该对象为止。即使内核重新自举了，该对象还是保持其值。Posix消息队列、信号量和共享内存区如果是使用映射文件实现的（不是必需条件），那么它们就是随文件系统持续的。

在定义一个IPC对象的持续性时我们必须小心，因为它并不总是像看起来的那样。例如管道内的数据是在内核中维护的，但管道具备的是随进程的持续性而不是随内核的持续性：最后

一个将某个管道打开着用于读的进程关闭该管道后，内核将丢弃所有的数据并删除该管道。类似地，尽管FIFO在文件系统中有名字，它们也只是具备随进程的持续性，因为最后一个将某个FIFO打开着的进程关闭该FIFO后，FIFO中的数据都被丢弃。

图1-3汇总了将在本书中讲述的各种类型IPC对象的持续性。

IPC类型	持续性
管道	随进程
FIFO	随进程
Posix互斥锁	随进程
Posix条件变量	随进程
Posix读写锁	随进程
fcntl记录上锁	随进程
Posix消息队列	随内核
Posix有名信号量	随内核
Posix基于内存的信号量	随进程
Posix共享内存区	随内核
System V消息队列	随内核
System V信号量	随内核
System V共享内存区	随内核
TCP套接字	随进程
UDP套接字	随进程
Unix域套接字	随进程

图1-3 各种类型IPC对象的持续性

注意该列表中没有任何类型的IPC具备随文件系统的持续性，但是我们说过有三种类型的Posix IPC可能会具备该持续性，这取决于它们的实现。显然，向一个文件写入数据提供了随文件系统的持续性，但这通常不作为一种IPC形式使用。多数形式的IPC并没有在系统重新自举后继续存在的打算，因为进程不可能跨越重新自举继续存活。对于一种给定形式的IPC，要求它具备随文件系统的持续性可能会使其性能降级，而IPC的一个基本的设计目标是高性能。

1.4 名字空间

当两个或多个无亲缘关系的进程使用某种类型的IPC对象来彼此交换信息时，该IPC对象必须有一个某种形式的名字（name）或标识符（identifier），这样其中一个进程（往往是服务器）可以创建该IPC对象，其余进程则可以指定同一个IPC对象。 7

管道没有名字（因此不能用于无亲缘关系的进程间），但是FIFO有一个在文件系统中的Unix路径名作为其标识符（因此可用于无亲缘关系的进程间）。在以后各章具体讲述其他形式的IPC时，我们将使用另外的命名约定。对于一种给定的IPC类型，其可能的名字的集合称为它的名字空间（name space）。名字空间非常重要，因为对于除普通管道以外的所有形式的IPC来说，名字是客户与服务器彼此连接以交换消息的手段。

图1-4汇总了不同形式的IPC所用的命名约定。

我们还指出哪些形式的IPC是由1996年版的Posix.1和Unix 98标准化的，这两个标准本身则在1.7节详细讨论。为了比较的目的，我们还包含了3种类型的套接字，它们在UNPv1中具体讲述。注意套接字API（应用程序编程接口）是由Posix.1g工作组标准化的，最终应该成为某个未来的Posix.1标准的一部分。

IPC类型	用于打开或创建IPC的名字空间	IPC打开后的标识	Posix.1 1996	Unix 98
管道	（没有名字）	描述符	•	•
FIFO	路径名	描述符	•	•
Posix互斥锁	（没有名字）	pthread_mutex_t指针	•	•
Posix条件变量	（没有名字）	pthread_cond_t指针	•	•
Posix读写锁	（没有名字）	pthread_rwlock_t指针		•
fcntl记录上锁	路径名	描述符	•	•
Posix消息队列	Posix IPC名字	mqd_t值	•	•
Posix有名信号量	Posix IPC名字	sem_t指针	•	•
Posix基于内存的信号量	（没有名字）	sem_t指针	•	•
Posix共享内存区	Posix IPC名字	描述符	•	•
System V消息队列	key_t键	System V IPC标识符		•
System V信号量	key_t键	System V IPC标识符		•
System V共享内存区	key_t键	System V IPC标识符		•
门	路径名	描述符		
Sun RPC	程序/版本	RPC句柄		
TCP套接字	IP地址与TCP端口	描述符	.1g	•
UDP套接字	IP地址与UDP端口	描述符	.1g	•
Unix域套接字	路径名	描述符	.1g	•

图1-4　各种形式IPC的名字空间

尽管Posix.1标准化了信号量，它们仍然是可选的特性。图1-5汇总了Posix.1和Unix 98对各种IPC特性的说明。每种特性有强制、未定义和可选三种选择。对于可选的特性，我们指出了其中每种特性受支持时（通常在<unistd.h>头文件中）定义的常值的名字，例如_POSIX_THREADS。

8

注意，Unix 98是Posix.1的超集。

IPC类型	Posix.1 1996	Unix 98
管道	强制	强制
FIFO	强制	强制
Posix互斥锁	_POSIX_THREADS	强制
Posix条件变量	_POSIX_THREADS	强制
进程间共享的互斥锁/条件变量	_POSIX_THREADS_PROCESS_SHARED	强制
Posix读写锁	（未定义）	强制
fcntl记录上锁	强制	强制
Posix消息队列	_POSIX_MESSAGE_PASSING	_XOPEN_REALTIME
Posix信号量	_POSIX_SEMAPHORES	_XOPEN_REALTIME
Posix共享内存区	_POSIX_SHARED_MEMORY_OBJECTS	_XOPEN_REALTIME
System V消息队列	（未定义）	强制
System V信号量	（未定义）	强制
System V共享内存区	（未定义）	强制
门	（未定义）	（未定义）
Sun RPC	（未定义）	（未定义）
mmap	_POSIX_MAPPED_FILE或_POSIX_SHARED_MEMORY_OBJECTS	强制
实时信号	_POSIX_REALTIME_SIGNALS	_XOPEN_REALTIME

图1-5　各种形式IPC的可用性