全国高等院校“十五”计算机规划教材
Text-Book for 21th Century on Network Technology

新世纪网络技术系列教材（1）

吴礼发　编著
谢希仁　审定

Programming Guide for Networks

网络程序设计教程

以实例描述具体的编程方法

具体应用方法与基本原理和方法的呼应教学

大量实训习题供实验上机练习

中国科学出版集团
北京希望电子出版社

Programming Guide for Networks 网络程序设计教程

全国高等院校“十五”计算机规划教材
Text-Book for 21th Century on Network Technology
新世纪网络技术系列教材（1）

吴礼发　编著
谢希仁　审定

Programming Guide for Networks

网络程序设计教程

以实例描述具体的编程方法

具体应用方法与基本原理和方法的呼应教学

大量实训习题供实验上机练习

中国科学出版集团
北京希望电子出版社

内 容 简 介

本教程是“新世纪网络技术系列教材”之一，该系列教材由网络技术主干课程教材组成，分别是《网络原理与技术教程》、《网络工程设计教程》、《网络程序设计教程》、《网络管理技术教程》、《网络安全技术教程》、《网络分布式计算》和《网络协议工程》。本系列教材可供本科、高职高专网络专业、计算机专业和相关 IT 专业根据网络课程的设置情况选用。

本教程介绍计算机网络程序设计的原理和方法，由四部分内容、共 9 章构成，主要内容包括计算机网络程序设计的概念和方法，Unix 系统下的基于 Berkeley 插口 API 的网络应用程序设计的原理和方法（本书的重点），Windows 环境下的基于 Windows 插口 API 的网络应用程序设计的原理和方法。此外，本书还介绍了另一种风格的计算机网络程序设计方法：VMS 系统下的 DECnet 网络程序设计的基本原理和方法。各章附有大量习题，可供上机练习。

本教程由高等院校具有丰富教学和开发经验的一线教师精心设计和撰写，在介绍各种具体的网络编程方法的同时力图清楚讲述带有共性的网络编程的原理和方法，同时，还比较透彻地分析了各种设计方法的原理以及各种异常情况的处理方法，强调学生学习后技术能力的提高和实现，强调具体编程方法的实现与基本原理方法的结合讨论，以提高学生对不同应用变化的应对能力。本教程体现了实践要求与教学目标的统一原则。

本教程可作为高校、高职计算机网络课的教科书、社会广大网络编程人员自学指导书和社会网络初、中级培训班教材。

本版 CD 含本教材实例源码。

系 列 盘 书：新世纪网络技术系列教材（1）
盘 书 名：网络程序设计教程 Programming Guide for Networks
总 策 划：北京希望电子出版社
文 本 著 作 者：吴礼发 编著 谢希仁 审定
C D 制 作 者：希望多媒体开发中心
C D 测 试 者：希望多媒体测试部
责 任 编 辑：马宏华 筱戎
出 版、发 行 者：北京希望电子出版社
地 址：北京中关村大街 26 号，100080
网址: www.bhp.com.cn E-mail: lwm@hope.com.cn
电话: 010-62562329, 62541992, 62637101, 62637102, 62633308, 62633309
（图书发行和技术支持）
010-62613322-215（门市） 010-62547735（编辑部）
经 销：各地新华书店、软件连锁店
排 版：希望图书输出中心 全卫
C D 生 产 者：北京中新联光盘有限责任公司
文 本 印 刷 者：北京双青印刷厂
开 本 / 规 格：787 毫米×1092 毫米 16 开本 15.75 印张 359 千字
版 次 / 印 次：2002 年 1 月第 1 版 2002 年 1 月第 1 次印刷
本 版 号：ISBN 7-900088-15-6
印 数：0001-5000
定 价：23.00 元（本版 CD）

新世纪网络技术系列教材

编　委　会

前　言

近几年来，以网络为基础的信息技术得到了快速的发展，各种网络应用大量地涌现出来。日益增长的网络应用需要大量的熟悉网络应用程序设计的人才。因此，给大学生开设计算机网络程序设计这门课程就变得越来越必要了。

网络编程是指利用网络编程接口来编写通过网络交换信息的应用程序。在大量的网络编程接口中，插口 API 是应用最广泛的，并且是 Unix 操作系统中最主要的网络编程接口。尽管 Windows 下的网络编程接口多种多样，但它们的基础仍然是插口 API。因此，本教材将重点放在 Unix 操作系统下的基于插口 API 的网络编程上。由于网络技术的飞速发展，以及不同平台上网络编程接口的差异，本教材在介绍具体的编程方法的同时，力图清楚地表述带有共性的网络编程的原理和方法，以适应这种变化，同时这也是作为教材的最重要的目的之一。

全书分为四部分，共 9 章。第一部分（第 1 章）是绪论，介绍计算机网络程序设计的一些基本内容、概念和方法。第二部分（第 2 章到第 7 章）主要讨论 Unix 系统下的网络编程。第 2 章是插口 API 简介，主要介绍 Berkeley 插口 API 中的基本概念、数据结构，接口函数的功能和使用方法，它们是基于插口 API 的网络应用程序设计的基础。第 3 章是 TCP 插口编程，主要介绍基于插口 API 的 TCP 插口编程概念和方法。第 4 章是 UDP 插口编程，主要介绍基于插口 API 的 UDP 插口编程概念和方法。第 5 章是基于插口的高级网络编程接口 NPORT 介绍，介绍一个简化的、方便的高级插口应用编程接口 NPORT。第 6 章是网络服务器的设计模式，讨论服务多客户的网络服务器的设计模式，在讨论网络服务器的设计模式之前还简单介绍了进程和线程的基本概念和编程接口。第 7 章是数据链路层的网络编程，介绍访问数据链路层服务的网络应用程序设计，并着重介绍了基于 libpcap 的网络编程方法。第三部分（第 8 章）主要阐述 Windows 环境下的网络编程，包括 Windows 插口 API 的基本概念、接口函数，以及 Windows 环境下的网络程序设计方法。第四部分（第 9 章）介绍另一种风格的网络应用编程：VMS 操作系统下的 DECnet 网络编程。

本教材可作为大学本科网络工程专业、计算机专业以及其他专业的计算机网络程序设计课程教材，参考学时为 50 至 60 学时。第 7 章和第 9 章的内容可作为选学内容。学习本门课程之前，读者最好已了解或掌握有关 C 语言程序设计、计算机网络、操作系统等课程的内容。因此，建议在大学四年级下学期开设本课程。本书也可作为网络研究和开发人员的自学教材和参考书。

计算机网络程序设计是一门实践性很强的课程，因此需要学习者通过大量的上机练习来理解网络编程概念和程序设计方法。书中配有大量实践性的习题可以选用。建议提供 20 个学时的上机实验课时。

在本书的编写过程中，谢希仁教授审阅了本书的全部内容，李旺、丁力、谢均、慎健等同志提供了部分有参考价值的素材并提出了许多宝贵意见，这里表示诚挚的谢意。由于时间仓促，加之编者水平有限，书中难免存在一些缺点和错误，希望广大读者批评指正。

吴礼发

wulifa@jlonline.com　2001 年 8 月于南京

全国高等院校计算机教材系列书

CX-2681
定价:16.00 元

CX-2682
定价:24.00 元

CX-2745
定价:19.00 元

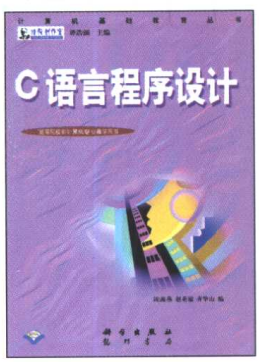
CX-2746
定价:14.00 元

CX-3250
定价: 26.00 元

CX-2934
定价:15.00 元

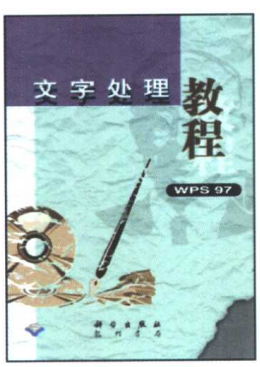
CX-2747
定价:23.00 元

CX-2749
定价:23.00 元



CX-3277
定价: 21.00 元

CX-2683
定价:15.00 元



CX-2684
定价:18.50 元

CX-2755
定价:27.00 元

CX-2756
定价:19.00 元

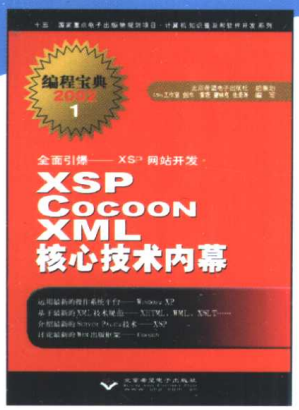
CX-2757
定价:16.00 元

CX-2758
定价:18.00 元

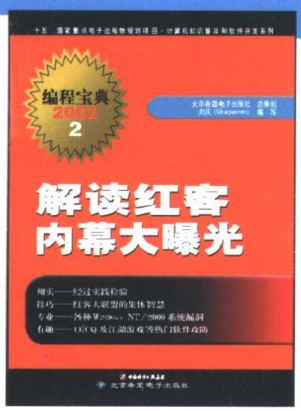
北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

地址: 北京中关村 083 信箱北京希望电子出版社 (邮编 100080)
电话: 010-62562329, 010-62633308 传真: 010-62579874
E-mail: grh@hope.com.cn

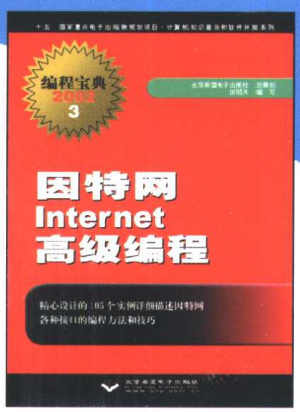
编程的利器、通向高级程序员的殿堂



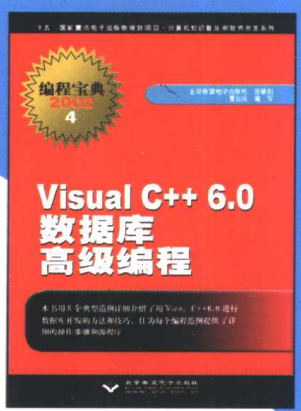
CX-83576
定价：58.00元



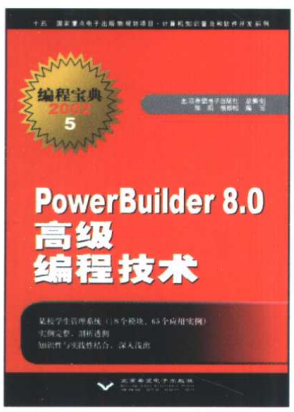
CX-83598
定价：30.00元



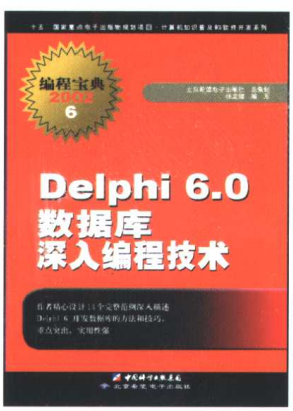
CX-83597
定价：55.00元



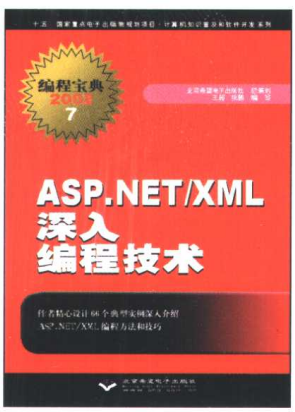
CX-83599
定价：35.00元



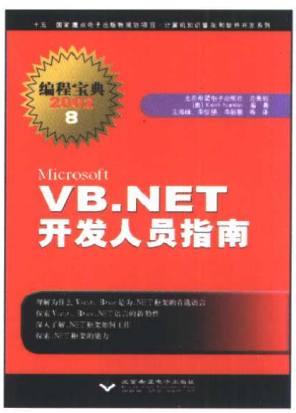
CX-83620
定价：42.00元



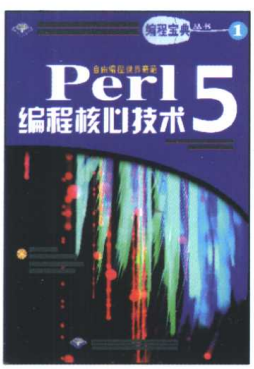
CX-83611
定价：35.00元



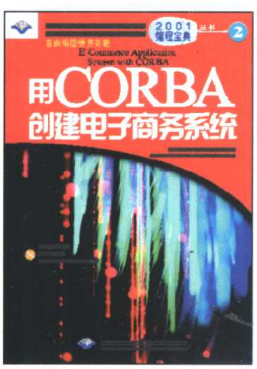
CX-83613
定价：35.00元



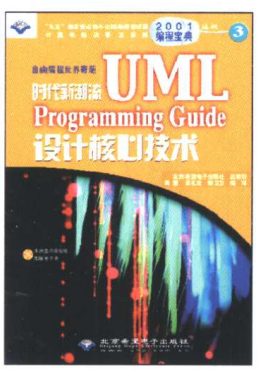
CX-83618
定价：33.00元



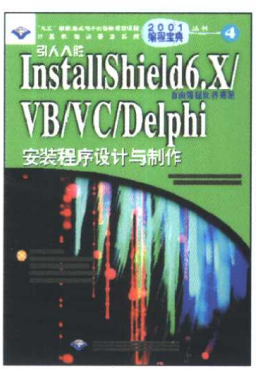
CX-83209
定价：48.00元



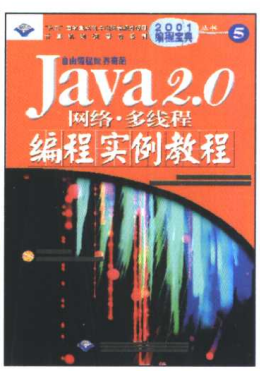
CX-83245
定价：30.00元



CX-83267
定价:35.00元



CX-83266
定价：50.00元



CX-83258
定价:39.00元

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

地址：北京中关村083信箱北京希望电子出版社（邮编 100080）
电话：010-62562329，010-62633308　传真：010-62579874
E-mail：grh@hope.com.cn

目　录

第1章 绪 论

本章介绍计算机网络程序设计的一些基本内容、概念和方法。

1-1 概 述

随着计算机网络的飞速发展和日益普及，网络应用越来越多。业界对计算机网络程序设计的需求也相应增多。因此，计算机网络程序设计作为一种重要的知识技能越来越受到人们的重视。

计算机网络程序设计就是利用网络应用编程接口编写网络应用程序，实现网络应用进程间的信息交互功能。

一般来说，应用进程间的通信可以分为两种：同一系统上的应用进程间的通信和不同系统上的应用进程间的通信。同一系统上的应用进程间的通信又称为进程间通信，即 IPC。而不同系统上的进程间的通信，必须通过网络编程接口访问网络协议提供的服务来实现。

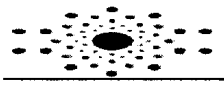
IPC 是进程间通信（Interprocess Communication）的简称。传统上该术语描述的是运行在某个操作系统上的不同进程间消息传递（Message Passing）的不同方式。IPC 的主要形式有：消息传递（管道、FIFO、消息队列），同步（互斥、条件变量、读写锁、文件与记录锁、信号灯），共享存储区（匿名共享存储区、有名共享存储区），远程过程调用（RPC）等。

本书主要讨论的是为实现不同系统上的进程间的通信而进行的网络应用编程的原理、接口和方法。事实上，同一系统上的不同应用进程间的通信也可以通过网络编程接口来实现，只是性能上会有些差别。很多网络应用编程接口，如插口（Socket，也有文献译为套接口或套接字，本书统一采用“插口”）API，对同一系统中的网络通信进行了优化（如通过 IPC）。考虑到分布式应用的可移植性等问题，使用网络编程接口比使用 IPC 机制要好。

网络通信离不开网络协议，网络编程接口访问网络协议所提供的服务。不同的网络协议可能提供不同的服务访问接口，同一网络应用编程接口可能提供访问不同网络协议的接口。如著名的网络应用编程接口——插口 API，支持对很多协议的访问，如 TCP，UDP，raw IP，数据链路层协议，Unix 域协议等。DECnet 网络应用编程接口则与插口完全不同，因为其协议的命名规则、地址格式、协议服务与 TCP/IP 有很大区别。然而，在一些系统中，插口 API 也支持 DECnet 协议。

Unix 域协议并不是一个实际的协议族，它只是在同一台主机上进行客户-服务器通信时，使用与不同主机上的客户和服务器间通信时采用的相同 API 的一种方法。

要学好网络编程，对于操作系统、网络协议、网络编程模式和方法要有比较好的理解，因为网络应用编程与它们是密不可分的。



1-2 网络服务

本书主要讨论基于 TCP/IP 协议栈的网络编程，即网络应用程序访问 TCP/IP 协议提供的服务来实现进程间的通信，并且主要是针对 IPv4 协议而不是 IPv6 协议。

网络协议是分层的，如 OSI/RM 分七层，TCP/IP 分四层。网络应用编程的目的是如何利用各协议层所提供的功能实现用户的应用，而不是要实现各协议层的功能。操作系统一般都把各协议层的功能作为系统内核的一部分来实现，应用程序只需调用系统提供的应用编程接口即可。

需要说明的是，因为协议的层次性，通过应用编程接口访问某一协议层提供的服务，事实上还需要间接用到其下各层提供的服务。

因为大多数应用程序访问运输层服务和数据链路层服务，因此本节主要介绍运输层服务和数据链路层服务。

1-2-1 运输层服务

从通信和信息处理的角度看，运输层属于面向通信部分的最高层。然而，从网络功能或用户功能来划分，则运输层又属于用户功能中的最低层。运输层的任务是根据下面通信子网的特性最佳地利用网络资源，并以可靠和经济的方式为两端主机上的进程之间透明地传送报文。正是因为运输层的这种特殊地位，大多数网络应用均要使用运输层提供的服务。因此，大部分网络应用程序设计都是针对运输层协议进行的。

TCP/IP 协议栈中的两个最主要的运输层协议是 TCP 和 UDP。其中，TCP 提供可靠的、有序的、端到端的数据传输服务，而 UDP 则提供的是不可靠的、不保证有序到达的、端到端的数据传输服务。

有关 TCP/IP 的文献非常之多，读者可以从中了解更多细节。

1-2-2 数据链路层服务

数据链路层负责在两个相邻结点间的链路上，无差错地传送以帧为单位的数据。每一帧包括一定数量的数据和一些必需的控制信息。数据链路层要负责建立、维护和释放数据链路的连接。在传送数据时，若接收结点检测到数据有差错，就要通知发方重发这一帧，直到这一帧正确到达接收结点为止。

目前，大多数的操作系统都提供了数据链路层访问接口，它使得应用程序可以实现以下功能：

（1）监视数据链路层上所收到的分组，这使得我们可以在普通计算机系统上通过 tcpdump 程序来监视系统，而无需特殊的硬件设备。

（2）作为普通应用程序而不是内核的一部分运行某些程序。例如，大多数 Unix 系统的 RARP 服务器是普通的应用进程，它们从数据链路读取 RARP 请求（RARP 不是 IP 数据报），并把应答写回数据链路。

异步转移模式（ATM）提供的也是数据链路层服务。在很多支持 ATM 的系统中（如 IBM 的 AIX 操作系统），也提供了类似插口 API 的访问 ATM 服务的应用编程接口。

1-3 网络应用编程接口

网络通信是进程间通信的一种非常重要的手段。网络通信是在协议的控制下进行的，网络编程的目的是访问网络协议提供的服务。访问协议提供的服务就是通过网络应用编程接口（API）来实现的。本节主要介绍一些流行的网络应用编程接口。

1-3-1 Berkeley 插口 API

插口 API，有时称为“Berkeley 插口 API”，因为它最开始出现在 Berkeley Unix 中。Berkeley 插口 API 是在 1983 年随 4.2 BSD 操作系统引入的，刚开始只支持 TCP/IP 协议族和 Unix 域协议。今天的插口还支持很多其他协议，如 OSI/RM 中的序列协议和 DECnet 协议等。

插口 API 是最重要的网络编程接口，特别是在 Unix 系统中，几乎所有的网络应用程序都是利用插口 API 来实现的。本书的重点就是讨论基于插口 API 的网络应用编程。

1-3-2 TLI

TLI 是 AT&T 开发的运输层接口。X/Open 组织于 1998 年发布了一个 TLI 的修正版 XTI（the X/Open Transport Interface，XTI/Open 传输接口）。XTI 只是对 TLI 作了少量修改，它是 TLI 的一个超集。

在上世纪 80 年代中期，Posix.1 出现之前，AT&T Unix 与 Berkeley Unix 之间存在着巨大差异。在当时的网络界，OSI/RM 协议的研究非常流行，并盛传 OSI/RM 协议将取代 TCP/IP。在这样的背景下，AT&T 于 1986 年在 System V 版本 3.0（SVR3）中首次推出称为 TLI（运输层接口）的不同于插口 API 的编程接口。TLI 与插口 API 编程有很多相似之处，不过 TLI 是按照 OSI/RM 传输服务定义建模的。

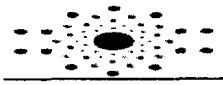
同插口 API 相比，使用 TLI 作为网络应用编程接口的程序设计人员要少得多。另外，TLI 编程方法与插口编程方法比较类似。因此，本书不准备介绍基于 TLI 的网络应用编程。

1-3-3 Windows Sockets

Windows Sockets（简称为 WinSock）API 是 Windows 下得到广泛应用的、开放的、支持多种协议的网络编程接口。从 1991 年的 1.0 版到 1997 年的 2.2.1 版，经过不断完善并在 Intel，Microsoft，Sun，SGI，Informix，Novell 等公司的全力支持下，已成为 Windows 环境下事实上的网络编程接口标准。

WinSock 以 Berkeley 插口 API 为范例定义了一套 Microsoft Windows 下网络编程接口。它不仅包含了人们所熟悉的 Berkeley Socket 风格的库函数，也包含了一组针对 Windows 的扩展库函数，以使程序员能充分地利用 Windows 消息驱动机制进行编程。

Windows Sockets 规范的目的在于提供给应用程序开发者一套简单的 API，并让各家网络软件供应商共同遵守。此外，在一个特定版本 Windows 的基础上，Windows Sockets 也定义了一个二进制接口（ABI），以此来保证使用 Windows Sockets API 的应用程序能够在任何网络软件供应商的符合 Windows Sockets 协议的实现上工作。因此，这份规范定义了应用程序开发者能够使用，并且网络软件供应商能够实现的一套库函数调用和相关语义。



由于 Windows 操作系统被广泛用于 PC 机，因此本书也将基于 Windows Sockets 的网络应用编程作为一个重要内容加以介绍。

1-3-4 可视化编程环境下的网络控件

随着 Microsoft Windows 下可视化编程工具的快速发展，如 Visual Basic，Visual C++，PowerBuilder，Delphi，C++ Builder 等，在这些可视化编程环境中的网络编程也变得越来越容易。在可视化编程环境中，大部分网络编程是通过网络控件来进行的。它屏蔽了网络编程细节，这同时也限制了网络编程的灵活性。

这些可视化网络控件也是基于 Windows Sockets 接口的，是比 Windows Sockets 更高一级的网络应用编程接口。

本书也将简要介绍这方面的内容。

1-3-5 其他网络编程接口

远程过程调用（RPC：Remote Procedure Call）指的是由本地系统上的进程激活远程系统上的进程，由远程过程完成某项任务后将结果返回给本地进程。RPC 在形式上与普通的本地过程调用类似，但其本质上的差异在于调用者与被调用者处在不同的主机系统上，因而需要解决数据表示、参数传递、调用语义、异常处理和安全等一系列问题。

RPC 最早由施乐(Xerox)公司提出，并于 1981 推出了第一个 RPC 产品——Courier RPC，该产品作为 XNS 的网络系统的一部分。后来，其他公司也推出了多种 RPC 的实现，如 Sun 公司的 RPC，Apollo 公司的 Apollo RPC 等。今天，RPC 成了大多数 Unix 操作系统上的标准配置。

RPC 是比插口 API 更高级的网络编程接口。RPC 程序设计包含两个层次：一个是高层 RPC 程序设计，另一个是低层 RPC 程序设计。对用户而言，高层 RPC 程序设计更少地涉及对运输层的控制以及鉴别机制的选择；而低层 RPC 程序设计则要由用户显式地进行运输层的控制和选择鉴别方式。

高层 RPC 编程接口只有三个库函数，且只能使用 UDP 作为传输协议。低层 RPC 编程接口的库函数就多得多，且可以选择 TCP 还是 UDP 作为传输协议，但需要程序员比较了解低层协议和 RPC 更多的细节内容。某些系统中有一些帮助进行低层 RPC 程序设计的工具，如 Sun RPC 中的 rpcgen 工具就可以帮助 RPC 程序员生成大部分的 RPC 程序，使程序员把主要精力放在客户方和服务器方的远程主程序设计上。

1-4 网络编程模式

网络编程模式一般采用客户-服务器模式（Client-Server）。客户-服务器模式描述的是进程之间的服务和被服务的关系。在这种模式下，大多数网络应用系统由两部分组成：客户（Client）和服务器（Server）。客户是主叫方，服务器是被叫方。

客户与服务器的通信关系一旦建立，通信就可以是双向的，客户和服务器都可以发送和接收信息。

在实际网络应用中，有很多客户服务器的例子，如 Web 浏览器（客户）和 Web 服务器

之间的信息交换；FTP 客户与 FTP 服务器之间的文件传输；Telnet 客户通过远程主机上的 Telnet 服务器提供远程登录手段。

一般情况下，客户与服务器之间的关系是多对多的关系，即一个客户可以与多个服务器通信，一个服务器可以同时与多个客户保持联系。根据实现方式的不同，服务器又可分为多种类型：串行服务器（单进程串行地服务客户请求），并发服务器（服务器为每一个客户创建一个进程或线程来服务客户的请求）。并发服务器又可分为：预先创建服务子进程并发服务器，预先创建服务线程并发服务器，按需创建服务子进程并发服务器，按需创建服务线程并发服务器等。还可以根据并发服务器中接受连接请求的方式对其作进一步区分。我们将在第 6 章中详细介绍服务器设计方法。

客户与服务器之间的通信涉及到网络通信协议。如，Web 浏览器客户与 Web 服务器之间的通信使用了 TCP 协议，而 TCP 又使用了下层的 IP 协议，IP 协议则使用下层的数据链路层协议，再往下则是物理层协议。因此，客户与服务器之间的通信可以这样来描述：在发送端，从上层往下层走，上层调用下层提供的服务，每一层次的协议的实现又为上一层提供服务；在接收端，从下层往上层走，执行相反的过程。

1-5 网络编程要考虑的问题

计算机网络程序设计因为涉及不同平台之间的信息交互，因此与单机上的程序设计有很大的差别。了解网络程序设计要注意的一些问题，有助于网络编程人员设计、编写高质量的网络应用程序。

1-5-1 并发环境下的网络编程

单进程应用与多进程或多线程应用程序的编程有着很大的区别。在多进程或多线程应用程序中，涉及到资源共享、进程或线程间的同步，因而要复杂得多。

在多进程或多线程应用中，使用的系统调用或函数必须是可重入的。哪些系统调用或系统函数是可重入的，这在不同系统中是不同的。一般的系统都会对此有详细的说明。对于那些不可重入的调用或函数，系统如果不提供多线程安全的版本，则应用编程人员需要避免使用或自己编写相应的函数。

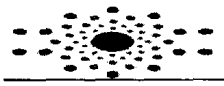
在多线程应用中，对调用或函数的使用有很多限制。如，在 Solaris2.5 中，在 Solaris 线程中访问定界数据会导致总线错误。在 Solaris 操作系统中，在一个线程中关闭另一个线程正在使用的网络端口会导致应用程序“死掉”，而这种情况在 Digital Unix（后称为 Tru64 Unix）中则不会出现。

因此，在多进程或多线程应用中，需要仔细考虑这些限制。

1-5-2 异构环境下的网络编程

网络通信常常是在多个平台之间进行，因此网络应用程序必须考虑不同平台之间的异构性，特别是不同平台上的数据格式的差异。这些差异主要表现在以下几个方面：

（1）**字节顺序**。不同的平台以不同的方式存放一个二进制数。最常见的有两种格式：大数在前的（big-endian）字节顺序和小数在前的（little-endian）字节顺序。大数在前的字



节顺序是指将一个多字节数的高序字节存储在内存的起始地址；而小数在前的字节顺序则相反，将低序字节存储在内存的起始地址。在操作系统中，IBM AIX，Sun OS，HP Unix，Solaris 采用大数在前的字节顺序；而 Digital Unix，Linux，BSDi，SystemV4，DOS，Windows 9x/2000/NT 则采用的是小数在前的字节顺序。同一数值在具有不同字节顺序的平台上的表示刚好相反。因此，作为网络编程人员，必须清楚各种字节顺序间的区别，并采用相应的措施来解决因这种差别所带来的问题。

网络应用程序中的通信数据主要包括两部分：网络分组中的首部和数据部分，首部与协议有关，数据部分与应用有关。不管是网络分组中的协议首部，还是网络分组中的数据部分都要考虑这个问题。协议首部将影响协议的正确操作，而分组中的数据则影响网络应用程序的功能。

（2）**字的长度**。不同的实现对于相同的数据类型可能有不同的表示长度。如，64 位操作系统与 32 位操作系统中，类型 long int 的长度是不一样的。在 Digital Unix 中的 long int 的长度为 8 字节，而在 Solaris 中则为 4 字节。对 short，int 或 long 等数据类型的大小也没有确定的规定。

（3）**字节定界问题**。不同的平台上给结构（struct）或联合（union）打包的方式也是不同的，这取决于所有数据类型的位数及机器的定界限制。一般情况下，操作系统在分配内存时，数据结构以 4 字节定界。例如，在很多系统中，默认情况下，结构 struct {char a; int b} 的长度为 8 而不是 5。只有在 1 字节定界的情况下其长度才为 5。

有几种方法可以解决上述问题。对于具有相同字节顺序的平台，通信双方均以单字节定界；对于具有不同字节顺序的平台间的通信，显式地定义格式（位数、字节顺序类型），远程过程调用（RPC）采用外部数据表示 XDR（External Data Representation），实际上就是用这种方法来解决这一问题；分布式计算平台：报文传递接口（MPI）也采取了类似的方法；另一种方法是，将需要发送的信息的结构在发送前变换成一种统一的格式（转换成一个字符数组），到达接收方后再执行相反的过程。对于数据结构中有比特变量的情况，处理起来更加复杂。因此，在实际网络编程中，尽量不要使用比特变量。

在很多网络协议的设计中，常常需要填充一些无用的字节以满足四字节定界，从而简化协议的实现。

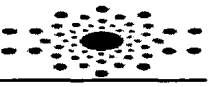
1-5-3 阻塞与非阻塞通信

可以将通信分为两种模式：阻塞和非阻塞。在网络编程时，选择通信模式也是一件很重要的事情。对于不同的协议，阻塞通信和非阻塞通信有不同的表现。

以插口为例，在阻塞模式下，利用 TCP 协议发送一个报文时，如果低层协议没有可用空间来存放用户数据，则应用进程将阻塞等待直到协议有可用的空间。而在非阻塞模式下，调用将直接返回而不需等待。在应用进程调用接收函数接收报文时，如果是在阻塞模式下，若没有到达的数据，则调用将一直阻塞直到有数据到达或出错；而在非阻塞模式下，将直接返回而不需等待。

对于 UDP 协议而言，因为 UDP 没有发送缓存，所有 UDP 协议即使在阻塞模式下也不会发生阻塞。

对于面向连接的协议，在连接建立阶段，阻塞与非阻塞也表现不一。在阻塞模式下，



如果没有连接请求到达，则等待连接调用将阻塞直到有连接请求到达；但在非阻塞模式下，如果没有连接请求到达，等待连接调用将直接返回。

在连接建立阶段，不管是阻塞模式还是非阻塞模式，发起连接请求的一方总是会使调用它的进程阻塞，阻塞间隔最少等于到达服务器的一次往返时间。

不同操作系统下，在非阻塞模式下，不能完成的 I/O 操作返回的错误代码也是不一样的。例如，系统 V 返回 EAGAIN 错误，而源自 Berkeley 的实现返回 EWOULDBLOCK 错误。更混乱的是，Posix.1 指定使用 EAGAIN 而 Posix.1g 指定使用 EWOULDBLOCK。幸运的是，大多数当前使用的系统（包括 SVR4 和 4.3BSD）将这两个错误代码定义为相同的值。

通信模式对应用程序的设计方法也有直接的影响。在非阻塞模式下，应用程序不断地轮询查看是否有数据到达或有连接请求到达。这种轮询方式比其他的技术耗费更多的 CPU 时间，因此要尽量避免使用，可以采用多路复用技术（调用 select 或 poll 函数）来解决这一问题。而在阻塞模式下，不存在这一问题。但是阻塞模式的缺点是进程或线程在执行 I/O 操作时将被阻塞而不能执行其他的工作，因此在单进程或单线程应用中不能使用这种模式。在多线程应用中比较适合采用阻塞模式，一个线程被阻塞不影响其他线程的工作。

另外，通信模式的选择与操作系统的类型也有关系。例如，在非占先的 Windows 操作系统中，应用程序应尽量不要使用阻塞模式工作。

1-5-4 服务类型的选择

从通信的角度看，网络协议栈中的各层所提供的服务可以分为两大类：面向连接（connection-oriented）服务与无连接（connectionless）服务。

（1）**面向连接服务**。所谓连接，就是两个对等实体为进行数据通信而进行的一种结合。面向连接服务要求：在数据交换之前，必须先建立连接；当数据交换结束后，则应终止这个连接。

一般来说，面向连接服务过程分为三个阶段：连接建立、数据传输和连接释放。在传送数据时是按序传送的。这点和电路交换的许多特性很相似，因此面向连接服务又称为“虚电路服务”。面向连接服务比较适合于在一段时间间隔内要向同一目的地发送许多报文的情况。对于发送很短的零星报文，连接建立和释放所带来的开销就显得过大了。

在 TCP/IP 协议栈中，TCP 协议提供面向连接的服务。

（2）**无连接服务**。无连接服务指的是两个实体之间的通信不需要先建立好一条连接，其所需的下层资源在数据传输时动态地进行分配。

无连接服务的另一个特征就是它不需要通信的两个实体同时是活跃的。只有发送端的实体正在进行发送时，它才必须是活跃的。而接收端的实体只有在进行接收操作时才必须是活跃的。

无连接服务的优点是灵活方便和效率高；但它不能防止报文的丢失、重复或失序，而将这些问题交由应用程序根据需要自行解决。无连接服务特别适合于传送少量零星的报文。

无连接服务又可分为以下三种类型：

■ 数据报（datagram）。它的特点是不需要接收端做任何响应，因而是一种不可靠的服务。数据报常被描述为“尽最大努力交付”（best effort delivery）。