

<<TCP/IP 网络互连技术>>

图书基本信息

书名：<<TCP/IP 网络互连技术>>

13位ISBN编号：9787302030942

10位ISBN编号：7302030944

出版时间：1998-10

出版时间：清华大学出版社

作者：科默

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<TCP/IP 网络互连技术>>

内容概要

内容简介

TCP/IP网络互连技术系列的第 卷讨论客户/服务器编程和应用，讲述了构筑所有分布式计算系统的客户/服务器计算模型的基本概念，内容涉及各种不同的服务器设计方法，以及用来构造客户/服务器的各种工具和技术，包括远程过程调用RPC。

书中包含了用来说明
每种设计和工具的运行程序示例的源代码。

卷 有三个

版本：BSD套接字版，AT&TTLI版，WindowsSockets版。

本书系WindowsSockets版，讨论在Internet上应用软件通过TCP/IP进行通信的问题，适用于Windows 95，WindowsNT，Win32环境下编程和使用Windows SocketAPI应用编程接口。

<<TCP/IP 网络互连技术>>

书籍目录

- Contents
- Foreword
- Preface
- Chapter 1 Introduction And Overview
 - 1.1 Use OfTCP/IP
 - 1.2 Designing Applications For A Distributed Environment
 - 1.3 Standard And Nonstandard Application Protocols
 - 1.4 An Example Of Standard Application Protocol Use
 - 1.5 An Example Connection
 - 1.6 Using TELNET To Access An Alternative Service
 - 1.7 Application Protocols And Software Flexibility
 - 1.8 Viewing Services From The Provider's Perspective
 - 1.9 The Remainder Of The Text
 - 1.10 Summary
- Chapter 2 The Client Server Model And Software Design
 - 2.1 Introduction
 - 2.2 Motivation
 - 2.3 Terminology And Concepts
 - 2.3.1 Clients And Servers
 - 2.3.2 Privilege And Complexity
 - 2.3.3 Standard Vs. Nonstandard Client Software
 - 2.3.4 Parameterization Of Clients
 - 2.3.5 Connectionless Vs. Connection-Oriented Servers
 - 2.3.6 Stateless Vs. Stateful Servers
 - 2.3.7 A Stateful File Server Example
 - 2.3.8 Statelessness Is A Protocol Issue
 - 2.3.9 Servers As Clients
 - 2.4 Summary
- Chapter 3 Concurrent Processing In Client-Server Software
 - 3.1 Introduction
 - 3.2 Concurrency In Networks
 - 3.3 Concurrency In Servers
 - 3.4 Terminology And Concepts
 - 3.4.1 The Process Concept
 - 3.4.2 Threads
 - 3.4.3 Programs vs. Threads
 - 3.4.4 Procedure Calls
 - 3.5 An Example Of Concurrent Thread Creation
 - 3.5.1 A Sequential C Example
 - 3.5.2 A Concurrent Version
 - 3.5.3 Timeslicing
 - 3.6 Diverging Threads
 - 3.7 Context Switching And Protocol Software Design
 - 3.8 Concurrency And Asynchronous I/O
 - 3.9 Concurrency Under UNIX

<<TCP/IP 网络互连技术>>

- 3.10 Execuling A Separately Compiled Program
- 3.11 Summary
- Chapter 4 Program Interface To Protocols
- 4.1 Introduction
- 4.2 Loosely Specified Protocol Software Interface
- 4.2.1 Advantages And Disadvantages
- 4.3 Interface Functionality
- 4.4 Conceptual Interface Specfication
- 4.5 Implementation Of An API
- 4.6 Two Basic Approaches To Network Communicatwn
- 4.7 The Basic I/O Functions Available In ANSI C
- 4.8 Hislory Of The UNIX Socket API
- 4.9 Summary
- Chapter 5 TheSocketAPI
- 5.1 Introduction
- 5.2 The History Of Sockets
- 5.3 Speifing A Protocol Interface
- 5.4 The Socket Ahstraction
- 5.4.1 Sockel Descriptors
- 5.4.2 System Data Structures For Sockets
- 5.4.3 Using Sockets
- 5.5 Specifying An Endpoint Address
- 5.6 A Generic Address Structure
- 5.7 Functions In The Sockel API
- 5.7.1 The WSAShutdown Function
- 5.7.2 The WSACleanup Function
- 5.7.3 The Socket Function
- 5.7.4 The Connect Function
- 5.7.5 The Send Function
- 5.7.6 The Recv Function
- 5.7.7 The Closesocket Function
- 5.7.8 The Bind Function
- 5.7.9 The Listen Function
- 5.7.10 The Accept Function
- 5.7.11 Summary Of Socket Calls Used Wilh TCP
- 5.8 Utility Roulines For Integer Conversion
- 5.9 Using Socket Culls In A Program
- 5.10 Symbolic Constants For Socket Call Parameters
- 5.11 Summary
- Chapter 6 Algorithms And Issues In Client Software Design
- 6.1 Introduction
- 6.2 Learning, Atgorilhms Instead Of Details
- 6.3 Client Architecture
- 6.4 Identifying The Location OfA Server
- 6.5 Parsing An Address Argumenl
- 6.6 Looking Up A Domain Name
- 6.7 Looking Up A Well-Known Port By Name

<<TCP/IP 网络互连技术>>

- 6.8 Port Numbers And Network Byte Order
- 6.9 Looking Up A Protocol By Name
- 6.10 The TCP Client Algorithm
- 6.11 Allocating A Socket
- 6.12 Choosing A Local Protocol Port Number
- 6.13 A Fundamental Problem In Choosing A Local IP Address
- 6.14 Connecting A TCP Socket To A Server
- 6.15 Communicating With The Server Using TCP
- 6.16 Reading A Response From A TCP Connection
- 6.17 Closing A TCP Connection
 - 6.17.1 The Need For Partial Close
 - 6.17.2 A Partial Close Operation
- 6.18 Programming A UDP Client
- 6.19 Connected And Unconnected UDP Sockets
- 6.20 Using Connect With UDP
- 6.21 Communicating With A Server Using UDP
- 6.22 Closing A Socket That Uses UDP
- 6.23 Partial Close For UDP
- 6.24 A Warning About UDP Unreliability
- 6.25 Summary
- Chapter 7 Example Client Software
 - 7.1 Introduction
 - 7.2 The Importance Of Small Examples
 - 7.3 Hiding Details
 - 7.4 An Example Procedure Library For Client Programs
 - 7.5 Implementation Of ConTCP
 - 7.6 Implementation Of ConUDP
 - 7.7 A Procedure That Forms Connections
 - 7.8 Using The Example Library
 - 7.9 The OAYTIME Service
 - 7.10 Implementation Of A TCP Client For DA YTIME
 - 7.11 Reading From A TCP Connection
 - 7.12 The TIME Service
 - 7.13 Accessing The TIME Service
 - 7.14 Accurate Times And Network Delays
 - 7.15 A UDP Client For The TIME Service
 - 7.16 The ECHO Service
 - 7.17 A TCP Client For The ECHO Service
 - 7.18 A UDP Client For The ECHO Service
 - 7.19 Summary
- Chapter 8 Algorithms And Issues In Server Software Design
 - 8.1 Introduction
 - 8.2 The Conceptual Server Algorithm
 - Concurrent Vs. Iterative Servers
 - Connection-Oriented Vs. Connectionless Access
 - Connection-Oriented Servers
 - Connectionless Servers

<<TCP/IP 网络互连技术>>

Failure, Reliability, And Statelessness
Optimizing Stateless Servers
Four Basic Types Of Servers
Request Processing Time
Iterative Server Algorithms
An Iterative, Connection-Oriented Server Algorithm
Binding To A Well-Known Address Using INADDR_ANY
Placing The Socket In Passive Mode
Accepting Connections And Using Them
An Iterative, Connectionless Server Algorithm
Forming A Reply Address In A Connectionless Server
Concurrent Server Algorithms
Master And Slave Threads
A Concurrent Connectionless Server Algorithm
A Concurrent, Connection-Oriented Server Algorithm
Using Separate Programs As Slaves
Apparent Concurrency Using A Single Thread
When To Use Each Server Type
A Summary of Server Types
The Important Problem Of Server Deadlock
Alternative Implementations
Summary
Chapter 9 Iterative, Connectionless Servers (UDP)
9.1 Introduction
9.2 Creating A Passive Socket
9.3 Thread Structure
9.4 An Example TIME Server
9.5 Summary
Chapter 10 Iterative, Connection-Oriented Servers (TCP)
10.1 Introduction
10.2 Allocating A Passive TCP Socket
10.3 A Server For The DAYTIME Service
10.4 Thread Structure
10.5 An Example DAYTIME Server
10.6 Closing Connections
10.7 Connection Termination And Server Vulnerability
10.8 Summary
Chapter 11 Concurrent, Connection-Oriented Servers (TCP)
11.1 Introduction
11.2 Concurrent ECHO
11.3 Iterative Vs. Concurrent Implementations
11.4 Thread Structure
11.5 An Example Concurrent ECHO Server
11.6 Summary
Chapter 12 Singly-Threaded, Concurrent Servers (TCP)
12.1 Introduction
12.2 Data-driven Processing In A Server

<<TCP/IP 网络互连技术>>

- 12.3 Data-Driven Processing With A Single Thread
- 12.4 Thread Structure Of A Singly-Threaded Server
- 12.5 An Example Singly-Threaded ECHO Server
- 12.6 Summary
- Chapter 13 Multiprotocol Servers (TCP, UDP)
- 13.1 Introduction
- 13.2 The Motivation For Reducing The Number Of Servers
- 13.3 Multiprotocol Server Design
- 13.4 Thread Structure
- 13.5 An Example Multiprotocol DAYTIME Server
- 13.6 The Concept Of Shared Code
- 13.7 Concurrent Multiprotocol Servers
- 13.8 Summary
- Chapter 14 Multiservice Servers (TCP, UDP)
- 14.1 Introduction
- 14.2 Consolidating Servers
- 14.3 A Connectionless, Multiservice Server Design
- 14.4 A Connection-Oriented, Multiservice Server Design
- 14.5 A Concurrent, Connection-Oriented, Multiservice Server
- 14.6 A Singly-Threaded, Multiservice Server Implementation
- 14.7 Invoking Separate Programs From A Multiservice Server
- 14.8 Multiservice, Multiprotocol Designs
- 14.9 An Example Multiservice Server
- 14.10 Static and Dynamic Server Configuration
- 14.11 An Example Super Server, Inetd
- 14.12 Summary
- Chapter 15 Uniform, Efficient Management Of Server Concurrency
- 15.1 Introduction
- 15.2 Choosing Between An Iterative And A Concurrent Design
- 15.3 Level Of Concurrency
- 15.4 Demand-Driven Concurrency
- 15.5 The Cost Of Concurrency
- 15.6 Overhead And Delay
- 15.7 Small Delays Can Matter
- 15.8 Thread Preallocation
- 15.8.1 Preallocation Techniques
- 15.8.2 Preallocation In A Connection-Oriented Server
- 15.8.3 Preallocation In A Connectionless Server
- 15.8.4 Preallocation, Bursty Traffic, And NFS
- 15.8.5 Preallocation On A Multiprocessor
- 15.9 Delayed Thread Allocation
- 15.10 The Uniform Basis For Both Techniques
- 15.11 Combining Techniques
- 15.12 Summary
- Chapter 16 Concurrency In Clients
- 16.1 Introduction
- 16.2 The Advantages Of Concurrency

<<TCP/IP 网络互连技术>>

- 16.3 The Motivation For Exercising Control
- 16.4 Concurrent Contact With Multiple Servers
- 16.5 Implementing Concurrent Clients
- 16.6 Singly-Threaded Implementations
- 16.7 An Example Concurrent Client That Uses ECHO
- 16.8 Execution Of The Concurrent Client
- 16.9 Managing A Timer
- 16.10 Example Output
- 16.11 Concurrency In The Example Code
- 16.12 Summary
- Chapter 17 Tunneling At The Transport And Application Levels
- 17.1 Introduction
- 17.2 Multiprotocol Environments
- 17.3 Mixing Network Technologies
- 17.4 Dynamic Circuit Allocation
- 17.5 Encapsulation And Tunneling
- 17.6 Tunneling Through An IP Internet
- 17.7 Application-Level Tunneling Between Clients And Servers
- 17.8 Tunneling, Encapsulation, And Dialup Phone Lines
- 17.9 Summary
- Chapter 18 Application Level Gateways
- 18.1 Introduction
- 18.2 Clients And Servers In Constrained Environments
 - 18.2.1 The Reality Of Multiple Technologies
 - 18.2.2 Computers With Limited Functionality
 - 18.2.3 Connectivity Constraints That Arise From Security
- 18.3 Using Application Gateways
- 18.4 Interoperability Through A Mail Gateway
- 18.5 Implementation Of A Mail Gateway
- 18.6 A Comparison Of Application Gateways And Tunneling
- 18.7 Application Gateways And Limited Functionality Systems
- 18.8 Application Gateways Used For Security
- 18.9 Application Gateways And The Extra Hop Problem
- 18.10 An Example Application Gateway
- 18.11 Details Of A Web-Based Application Gateway
- 18.12 Invoking A CGI Program
- 18.13 URLs For The RFC Application Gateway
- 18.14 A General-Purpose Application Gateway
- 18.15 Operation Of SLIRP
- 18.16 How SLIRP Handles Connections
- 18.17 IP Addressing And SURP
- 18.18 Summary
- Chapter 19 External Data Representation (XDR)
- 19.1 Introduction
- 19.2 Representations For Data In Computers
- 19.3 The N-Squared Conversion Problem
- Network Standard Byte Order

<<TCP/IP 网络互连技术>>

A De Facto Standard External Data Representation

XDR Data Types

Implicit Types

Software Support For Using XDR

XDR Library Routines

Building A Message One Piece At A Time

Conversion Routines In The XDR Library

XDR Streams, 1/0, and TCP

Records, Record Boundaries, And Datagram 1/0

Summary

Chapter 20, Remote Procedure Call Concept (RPC)

20.1 Introduction

20.2 Remote Procedure Call Model

20.3 Two Paradigms For Building Distributed Programs

20.4 A Conceptual Model For Conventional Procedure Calls

20.5 An Extension Of the Procedural Model

20.6 Execution Of Conventional Procedure Call And Return

20.7 The Procedural Model In Distributed Systems

20.8 Analogy Between Client-Server And RPC

20.9 Distributed Computation As A Program

20.10 Sun Microsystems' Remote Procedure Call Definition

20.11 Remote Programs And Procedures

20.12 Reducing The Number Of Arguments

20.13 Identifying Remote Programs And Procedures

20.14 Accommodating Multiple Versions Of A Remote Program

20.15 Mutual Exclusion For Procedures In A Remote Program

20.16 Communication Semantics

20.17 At Least Once Semantics

20.18 RPC Retransmission

20.19 Mapping A Remote Program To A Protocol Port

20.20 Dynamic Port Mapping

20.21 RPC Port Mapper Algorithm

20.22 RPC Message Format

20.23 Marshaling Arguments For A Remote Procedure

20.24 Authentication

20.25 An Example Of RPC Message Representation

20.26 An Example Of An Authentication Field

20.27 Summary

Chapter 21 Distributed Program Generation (Rpcgen Concept)

21.1 Introduction

21.2 Using Remote Procedure Calls

21.3 Programming Mechanisms To Support RPC

21.4 Dividing A Program Into Local And Remote Procedures

21.5 Adding Code For RPC

21.6 Stub Procedures

21.7 Multiple Remote Procedures And Dispatching

21.8 Name Of The Client-Side Stub Procedure

<<TCP/IP 网络互连技术>>

- 21.9 Using Rpcgen To Generate Distributed Programs
- 21.10 Rpcgen Output And Interface Procedures
- 21.11 Rpcgen Input And Output
- 21.12 Using Rpcgen To Build A Client And Server
- 21.13 Summary
- Chapter 22 Distributed Program Generation (Rpcgen Example)
- 22.1 Introduction
- 22.2 An Example To Illustrate Rpcgen
- 22.3 Dictionary Look Up
- 22.4 Eight Steps To A Distributed Application
- 22.5 Step 1: Build A Conventional Application Program
- 22.6 Step 2: Divide The Program Into Two Parts
- 22.7 Step 3: Create An Rpcgen Specification
- 22.8 Step 4: Run Rpcgen
- 22.9 The h File Produced By Rpcgen
- 22.10 The XDR Conversion File Produced By Rpcgen
- 22.11 The Client Code Produced By Rpcgen
- 22.12 The Server Code Produced By Rpcgen
- 22.13 Step 5: Write Stub Interface Procedures
- 22.13.1 Client-Side Interface Routines
- 22.13.2 Server-Side Interface Routines
- 22.14 Step 6: Compile And Link The Client Program
- 22.15 Step 7: Compile And Link The Server Program
- 22.16 Step 8: Start The Server And Execute The Client
- 22.17 Summary
- Chapter 23 Network File System Concepts (NFS)
- 23.1 Introduction
- 23.1 Remote File Access Vs. Transfer
- 23.3 Operations On Remote Files
- 23.4 File Access Among Heterogeneous Computers
- 23.5 Stateless Servers
- 23.6 NFS And UNIX File Semantics
- 23.7 Review Of The UNIX File System
- 23.7.1 Basic Definitions
- 23.7.2 A Byte Sequence Without Record Boundaries
- 23.7.3 A File 's Owner And Group Identifiers
- 23.7.4 Protection And Access
- 23.7.5 The UNIX Open-Read-Write-Close Paradigm
- 23.7.6 UNIX Data Transfer
- 23.7.7 Permission To Search A Directory
- 23.7.8 UNIX Random Access
- 23.7.9 Seeking Beyond The End Of A UNIX File
- 23.7.10 UNIX File Position And Concurrent Access
- 23.7.11 Semantics Of Write During Concurrent Access
- 23.7.12 UNIX File Names And Paths
- 23.7.13 The UNIX tnode: Information Stored With A File
- 23.7.14 The UNIX Stat Operation

<<TCP/IP 网络互连技术>>

- 23.7.15 The UNIX File Naming Mechanism
- 23.7.16 UNIX File System Mounts
- 23.7.17 UNIX File Name Resolution
- 23.7.18 UNIX Symbolic Links
- 23.8 Files Under NFS
- 23.9 NFS File Types
- 23.10 NFS File Modes
- 23.11 NFS File Attributes
- 23.12 NFS Client And Server
- 23.13 NFS Client Operation
- 23.14 NFS Client And UNIX
- 23.15 NFS Mounts
- 23.16 File Handle
- 23.17 NFS Handles Replace Path Names
- 23.18 An NFS Client Under Windows
- 23.19 File Positioning With A Stateless Server
- 23.20 Operations On Directories
- 23.21 Reading A Directory Slatelessly
- 23.22 Mulliple Hierarchies In An NFS Server
- 23.23 The Mount Protocol
- 23.24 Summary
- Chapter 24 Network File System Protocol (NFS, Mount)
- 24.1 Inltroduction
- 24.2 Using RPC To Define A Protocol
- 24.3 Defining A Protocol With Data Structures And Procedures
- 24.4 NFS Conslantl, Type, And Data Declarations
- 24.4.1 NFSConstants
- 24.4.2 NFS Typedef Declarations
- 24.4.3 NFS Data Structures
- 24.5 NFS Procedures
- 24.6 Semantics Of NFS Operations
- 24.6.1 NFSPROC_NULL (Procedure 0)
- 24.6.2 NFSPROC_GETA TTR (Procedure 1)
- 24.6.3 NFSPROC_SETATTR (Procedure 2)
- 24.6.4 NFSPROC_ROOT (Procedure 3) [Ohsolete in NFS3]
- 24.6.5 NFSPROC_OOKUP (Procedure 4)
- 24.6.6 NFSPROC_READLINK (Procedure 5)
- 24.6.7 NFSPROC_READ (Procedure 6)
- 24.6.8 NFSPROC_WRITECACHE (Procedure 7) [Obsolele in NFS3]
- 24.6.9 NFSPROC_WRITE (Procedure 8)
- 24.6.10 NFSPROC_REATE (Procedure 9)
- 24.6.11 NFSPROC_REMOVE (Procedure 10)
- 24.6.12 NFSPROC_RENAME (Procedure 11)
- 24.6.13 NFSPROC_LINK (Procedure 12)
- 24.6.14 NFSPROC_SYMUNK (Procedure 13)
- 24.6.15 NFSPROC_MKDIR (Procedure 14)
- 24.6.16 NFSPROC_RMD1R (Procedure 15)

<<TCP/IP 网络互连技术>>

- 24.6.17 NFSPROC_READDIR (Procedure 16)
- 24.6.18 NFSPROC_STATFS (Procedure 17)
- 24.7 The Mount Protocol
 - 24.7.1 Mount Constant Definitions
 - 24.7.2 Mount Type Definitions
 - 24.7.3 Mount Data Structures
- 24.8 Procedures In The Mount Protocol
- 24.9 Semantics of Mount Operations
 - 24.9.1 MNTPROC_ULL (Procedure 0)
 - 24.9.2 MNTPROC_MNT (Procedure 1)
 - 24.9.3 MNTPROC_DUMP (Procedure 2)
 - 24.9.4 MNTPROC_MNT (Procedure 3)
 - 24.9.5 MNTPROC_UMNTALL (Procedure 4)
 - 24.9.6 MNTPROC_EXPORT (Procedure 5)
- 24.10 NFS And Mount Authentication
- 24.11 Changes In NFS Version 3
- 24.12 Summary
- Chapter 25 A TELNET Client (Program Structure)
- 25.1 Introduction
- 25.2 Overview
 - 25.2.1 The User's Terminal
 - 25.2.2 Command And Control Information
 - 25.2.3 Terminals, Windows, and Files
 - 25.2.4 The Need For Concurrency
 - 25.2.5 A Thread Model For A TELNET Client
- 25.3 A TELNET Client Algorithm
- 25.4 Keyboard I/O In Windows
- 25.5 Global Variables Used For Keyboard Control
- 25.6 Initializing The Keyboard Thread
- 25.7 Finite State Machine Specification
- 25.8 Embedding Commands In A TELNET Data Stream
- 25.9 Option Negotiation
- 25.10 Request/Offer Symmetry
- 25.11 TELNET Character Definitions
- 25.12 A Finite State Machine For Data From The Server
- 25.13 Transitions Among States
- 25.14 A Finite State Machine Implementation
- 25.15 A Compact FSM Representation
- 25.16 Keeping The Compact Representation At Run-Time
- 25.17 Implementation Of A Compact Representation
- 25.18 Building An FSM Transition Matrix
- 25.19 The Socket Output Finite State Machine
- 25.20 Definitions For The Socket Output FSM
- 25.21 The Option Subnegotiation Finite State Machine
- 25.22 Definitions For The Option Subnegotiation FSM
- 25.23 FSM Initialization
- 25.24 Arguments For The TELNET Client

<<TCP/IP 网络互连技术>>

- 25.25 The Heart Of The TELNET Client
- 25.26 TELNET Synchronization
- 25.27 Handling A Severe Error
- 25.28 Implementation Of The Main FSM
- 25.29 A Procedure For Immediate Disconnection
- 25.30 Abort Procedure
- 25.31 Summary
- Chapter 26 A TELNET Client (Implementation Details)
- 26.1 Introduction
- 26.2 The FSM Action Procedures
- 26.3 Recording The Type Of An Option Request
- 26.4 Performing No Operation
- 26.5 Responding To WILLWONT For The Echo Option
- 26.6 Sending A Response
- 26.7 Responding To WILUWONT For Unsupported Options
- 26.8 Responding To WLLWONT For The No Go-Ahead Option
- 26.9 Generating DO/DONT For Binary Transmission
- 26.10 Responding To DO/DONT For Unsupported Options
- 26.11 Responding To DO/DONT For Transmit Binary Option
- 26.12 Responding To DO/DONT For The Terminal Type Option
- 26.13 Option Subnegotiation
- 26.14 Sending Terminal Type Information
- 26.15 Terminating Subnegotiation
- 26.16 Sending A Character To The Server
- 26.17 Displaying Incoming Data On The User's Terminal
- 26.18 Writing A Block Of Data To The Server
- 26.19 Interacting With The Local Client
- 26.20 Responding To Illegal Commands
- 26.21 Scripting To A File
- 26.22 Implementation Of Scripting
- 26.23 Initialization Of Scripting
- 26.24 Collecting Characters Of The Script File Name
- 26.25 Opening A Script File
- 26.26 Terminating Scripting
- 26.27 Printing Status Information
- 26.28 Summary
- Chapter 27 Porting Servers From UNIX To Windows
- 27.1 Introduction
- 27.2 Operating in Background
- 27.3 Shared Descriptors And Inheritance
- 27.4 The Controlling TTY
- 27.5 Working Directories
- 27.6 File Creation And Umask
- 27.7 Process Groups
- 27.8 Descriptors For Standard I/O
- 27.9 Mutual Exclusion For A Server
- 27.10 Recording A Process ID

<<TCP/IP 网络互连技术>>

- 27.11 Waiting For A Child Process To Exit
- 27.12 Using A System Log Facility
 - 27.12.1 Generating Log Messages
- 27.13 Miscellaneous Incompatibilities
- 27.14 Summary
- Chapter 28 Deadlock And Starvation In Client-Server Systems
 - 28.1 Introduction
 - 28.2 Definition Of Deadlock
 - 28.3 Difficulty Of Deadlock Detection
 - 28.4 Deadlock Avoidance
 - 28.5 Deadlock Between A Client And Server
 - 28.6 Avoiding Deadlock In A Single Interaction
 - 28.7 Starvation Among A Set Of Clients And A Server
 - 28.8 Busy Connections And Starvation
 - 28.9 Avoiding Blocking Operations
 - 28.10 Threads, Connections, And Other Limits
 - 28.11 Cycles Of Clients And Servers
 - 28.12 Documenting Dependencies
 - 28.13 Summary
- Appendix 1 Functions And Library Routines Used With Sockets
- Appendix 2 Manipulation Of Windows Socket Descriptors
- Bibliography
- Index

<<TCP/IP 网络互连技术>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介, 请支持正版图书。

更多资源请访问:<http://www.tushu007.com>