

图书基本信息

书名：<<TCP/IP 网络互连(第3卷):客户/服务器编程及应 (平装)>>

13位ISBN编号：9787115099211

10位ISBN编号：7115099219

出版时间：2002-1

出版时间：人民邮电出版社

作者：科默

页数：599

字数：766000

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>

内容概要

本书讨论了客户/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，内容包括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，包括远程调用RPC。

书中包括了用来说明各种设计和工具的运行程序示例的源代码。

这本书是基于Linux/POSIX Sockets版本编写的，组织结构合理，易于阅读，是一本关于TCP/IP网络互连的既经典又可读性极强的书，是任何一个想要了解网络互连技术的人所必不可少的参考书。

本书适合作为高等院校计算机专业网络相关课程的教材，也适合各类网络技术开发人员阅读。

书籍目录

Chapter 1 Introduction And Overview 1.1 Internet Applications Using TCP/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 TELNET 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 This 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 Stateless File Server Example 2.3.8 A Stateful File Server Example 2.3.9 Identifying A Client 2.3.10 Statelessness Is A Protocol Issue 2.3.11 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 Sharing Of Local And Global Variables 3.4.3 Procedure Calls 3.5 An Example Of Concurrent Process Creation 3.5.1 A Sequential C Example 3.5.2 A Concurrent Version 3.5.3 Timeslicing 3.5.4 Singly-Threaded Process Assumption 3.5.5 Making Processes Diverge 3.6 Executing New Code 3.7 Context Switching And Protocol Software Design 3.8 Concurrency And Asynchronous I/O 3.9 Summary Chapter 4 Application 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 Specification 4.5 System Calls 4.6 Two Basic Approaches To Network Communication 4.7 The Basic I/O Functions Available In Linux 4.8 Using Linux I/O With TCP/IP 4.9 Summary Chapter 5 The Socket API 5.1 Introduction 5.2 Berkeley Sockets 5.3 Specifying A Protocol Interface 5.4 The Socket Abstraction 5.4.1 Socket Descriptors And File Descriptors 5.4.2 System Data Structures For Sockets 5.4.3 Making A Socket Active Or Passive 5.5 Specifying An Endpoint Address 5.6 A Generic Address Structure 5.7 Major System Calls In The Socket API 5.7.1 The Socket Call 5.7.2 The Connect Call 5.7.3 The Send Call 5.7.4 The Recv Call 5.7.5 The Close Call 5.7.6 The Bind Call 5.7.7 The Listen Call 5.7.8 The Listen Call 5.7.9 Using Read And Write With Sockets 5.7.10 Summary Of Socket Calls 5.8 Utility Routines For Integer Conversion 5.9 Using Socket Calls 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 Algorithms Instead Of Details 6.3 Client Architecture 6.4 Identifying The Location Of A Server 6.5 Parsing An Address Argument 6.6 Looking Up A Domain Name 6.7 Looking Up A Well-Known Port By Name 6.8 Port Numbers And Network Byte Order 6.9 Looking Up A Protocol By Name 6.10 The TCP Client Algorithm 6.11 Allocation 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 Receiving 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 ConnectTCP 7.6 Implementation Of ConnectUDP 7.7 A Procedure That Forms Connections 7.8 Using The Example Library 7.9 The DAYTIME Service 7.10 Implementation Of A TCP Client For DAYTIME 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 8.3 Concurrent Vs. Iterative Servers 8.4 Connection-Oriented Vs. Connectionless Access 8.5 Transport Protocol Semantics 8.5.1 TCP Semantics 8.5.2 UDP Semantics 8.6 Choice Of Transport 8.7 Connection-Oriented Servers 8.8 Connectionless Servers 8.9 Failure , Reliability , And Statelessness

8.10 Optimizing Stateless Servers 8.11 Four Basic Types Of Servers 8.12 Request Processing Time 8.13 Iterative Server Algorithms 8.14 An Iterative , Connection-Oriented Server Algorithm 8.15 Binding To A Well-Known Address Using INADDR_ANY 8.16 Placing The Socket In Passive Mode 8.17 Accepting Connections And Using Them 8.18 An Iterative , Connectionless Server Algorithm 8.19 Forming A Reply Address In A Connectionless Server 8.20 Concurrent Server Algorithms 8.21 Master And Slaves 8.22 A Concurrent , Connectionless Server Algorithm 8.23 A Concurrent , Connection-Oriented Server Algorithm 8.24 Implementations Of Server Concurrency 8.25 Using Separate Programs As Slaves 8.26 Apparent Concurrency Using A Single Thread 8.27 When To Use Each Server Type 8.28 A Summary of Server Types 8.29 The Important Problem Of Server Deadlock 8.30 Alternative Implementations 8.31 Summary Chapter 9 Iterative , Connectionless Servers(UDP) 9.1 Introduction 9.2 Creating A Passive Socket 9.3 Process Structure 9.4 An Example TIME Server 9.5 Summary Chapter 10 Iterative , Connection-Oriented Servers(TCP) 10.1 Introduction 10.2 Allocation A Passive TCP Socket 10.3 A Server For The DAYTIME Service 10.4 Process 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 ECHO Service 11.3 Iterative Vs. Concurrent Implementations 11.4 Process Structure 11.5 An Example Concurrent ECHO Server 11.6 Cleaning Up Errant Processes 11.7 Summary Chapter 12 Using Threads For Concurrency(YCP) 12.1 Introduction 12.2 Overview Of Linux Threads 12.3 Advantages Of Threads 12.5 Descriptors , Delay , And Exit 12.6 Thread Exit 12.7 Thread Coordination And Synchronization 12.7.1 Mutex 12.7.2 Semaphore 12.7.3 Condition Variable 12.8 An Example Server Using Threads 12.9 Monitor And Control 12.10 Summary Chapter 13 Single-Thread , Concurrent Servers(TCP) 13.1 Introduction 13.2 Data-driven Processing In a Server 13.3 Data-Driven Processing With A Single Thread 13.4 Process Structure Of A Single-Thread Server 13.5 An Example Single-Thread ECHO Server 13.6 Summary Chapter 14 Multiprotocol Servers(TCP , UDP) 14.1 Introduction 14.2 The Motivation For Reducing The Number Of Servers 14.3 Multiprotocol Server Design 14.4 Process Structure 14.5 An Example Multiprotocol DAYTIME Server 14.6 The Concept Of Shared Code 14.7 Concurrent Multiprotocol Servers 14.8 Summary Chapter 15 Multiservice Servers(TCP , UDP) 15.1 Introduction 15.2 Consolidation Servers 15.3 A Connectionless , Multiservice Server Design 15.4 A Connection-Oriented , Multiservice Server Design 15.5 A Concurrent , Connection-Oriented , Multiservice Server 15.6 A Single-Thread , Multiservice Server Implementation 15.7 Invoking Separate Programs From A Multiservice Server 15.8 Multiservice , Multiprotocol Designs 15.9 An Example Multiservice Server 15.10 Static and Dynamic Server Configuration 15.11 The Super Server , Inetd 15.12 An Example Inetd Server 15.13 A List Of Server Variations 15.14 Summary Chapter 16 Uniform , Efficient Management Of Server Concurrency 16.1 Introduction 16.2 Choosing Between An Iterative And A Concurrent Design 16.3 Level Of Concurrency 16.4 Demand-Driven Concurrency 16.5 The Cost Of Concurrency 16.6 Overhead And Delay 16.7 Small Delays Can Matter 16.8 Slave Preallocation 16.8.1 Preallocation In Linux 16.8.2 Preallocation In A Connection-Oriented Server 16.8.3 Mutex , File Locking , and Concurrent Calls To Accept 16.8.4 Preallocation In A Connectionless Server 16.8.5 Preallocation , Bursty Traffic , And NFS 16.8.6 Preallocation On A Multiprocessor 16.9 Delayed Slave Allocation 16.10 The Uniform Basis For Both Techniques 16.11 Combining Techniques 16.12 Summary Chapter 17 Concurrency In Clients 17.1 Introduction 17.2 The Advantages Of Concurrency 17.3 The Motivation For Exercising Control 17.4 Concurrent Contact With Multiple Servers 17.5 Implementing Concurrent Clients 17.6 Single-Thread Implementations 17.7 An Example Concurrent Client That Uses ECHO 17.8 Execution Of The Concurrent Client 17.9 Concurrency In The Example Code 17.10 Summary Chapter 18 Tunneling At The Transport And Application Levels 18.1 Introduction 18.2 Multiprotocol Environments 18.3 Mixing Network Technologies 18.4 Dynamic Circuit Allocation 18.5 Encapsulation And Tunneling 18.6 Tunneling Through An IP Internet 18.7 Application-Level Tunneling Between Clients And Servers 18.8 Tunneling , Encapsulation , And Dialup Phone Lines 18.9 Summary Chapter 19 Application Level Gateways 19.1 Introduction 19.2 Clients And Servers In Constrained Environments 19.2.1 The Reality Of Limited Access 19.2.2 Computers With Limited Functionality 19.2.3 Connectivity Constraints That Arise From Security 19.3 Using Application Gateways 19.4

Interoperability Through A Mail Gateway 19.5 Implementation Of A Mail Gateway 19.6 A Comparison Of Application Gateways And Tunneling 19.7 Application Gateways And Limited Internet Connectivity 19.8 Application Gateways Used For Security 19.9 Application Gateways And The Extra Hop Problem 19.10 An Example Application Gateway 19.11 Implementation Of An Application Gateway 19.12 Code For The Application Gateway 19.13 An Example Gateway Exchange 19.14 Using Rfcd With .forward Or Slocal 19.15 A General-Purpose Application Gateway 19.16 Operation Of SLIRP 19.17 How SLIRP Handles Connections 19.18 IP Addressing And SLIRP 19.19 Summary Chapter 20 External Data Representation(XDR) 20.1 Introduction 20.2 Representations Of Data 20.3 Asymmetric Conversion And The N-Squared Problem 20.4 Network Standard Byte Order 20.5 A De Facto Standard External Data Representation 20.6 XDR Data Types 20.7 Implicit Types 20.8 Software Support For Using XDR 20.9 XDR Library Routines 20.10 Building A Message One Piece At A time 20.11 Conversion Routines In The XDR Library 20.12 XDR Streams , I/O , and TCP 20.13 Records , Record Boundaries , And Datagram I/O 20.14 Summary Chapter 21 Remote Procedure Call Concept(RPC) 21.1 Introduction 21.2 Remote Procedure Call Model 21.3 Two Paradigms For Building Distributed Programs 21.4 A Conceptual Model For Conventional Procedure Calls 21.5 An Extension Of the Procedural Model 21.6 Execution Of Conventional Procedure Call And Return 21.7 The Procedural Model In Distributed Systems 21.8 Analogy Between Client-Server And RPC 21.9 Distributed Computation As A Program 21.10 Sun Microsystems' Remote Procedure Call Definition 21.11 Remote Programs And Procedures 21.12 Reducing The Number Of Arguments 21.13 Identifying Remote Programs And Procedures 21.14 Accommodating Multiple Versions Of A Remote Program 21.15 Mutual Exclusion For Procedures In A Remote Program 21.16 Communication Semantics 21.17 At Least Once Semantics 21.18 RPC Retransmission 21.19 Mapping A Remote Program To A Protocol Port 21.20 Dynamic Port Mapping 21.21 RPC Port Mapper Algorithm 21.22 ONC RPC Message Format 21.23 Marshaling Arguments For A Remote Procedure 21.24 Authentication 21.25 An Example Of RPC Message Representation 21.26 An Example Of The UNIX Authentication Field 21.27 Summary Chapter 22 Distributed Program Generation(Rpcgen Concept) 22.1 Introduction 22.2 Using Remote Procedure Calls 22.3 Programming Mechanisms To Support RPC 22.4 Dividing A Program Into Local And Remote Procedures 22.5 Adding Code For RPC 22.6 Stub Procedures 22.7 Multiple Remote Procedures And Dispatching 22.8 Name Of The Client-Side Stub Procedure 22.9 Using Rpcgen To Generate Distributed Programs 22.10 Rpcgen Output And Interface Procedures 22.11 Rpcgen Input And Output 22.12 Using Rpcgen To Build A Client And Server 22.13 Summary Chapter 23 Distributed Program Generation(Rpcgen Example) 23.1 Introduction 23.2 An Example To Illustrate Rpcgen 23.3 Dictionary Operations 23.4 Eight Steps To A Distributed Application 23.5 Step1 : Build A Conventional Application Program 23.6 Step2 : Divide The Program Into Two Parts 23.7 Step3 : Create An Rpcgen Specification 23.8 Step4 : Run Rpcgen 23.9 The .h File Produced By Rpcgen 23.10 The XDR Conversion File Produced By Rpcgen 23.11 The Client Code Produced By Rpcgen 23.12 The Server Code Produced By Rpcgen 23.13 Step5 : Write Stub Interface Procedures 23.13.1 Client-Side Interface Routines 23.13.2 Server-Side Interface Routines 23.14 Step6 : Compile And Link The Client Program 23.15 Step7 : Compile And Link The Server Program 23.16 Step8 : Start The Server And Execute The Client 23.17 Using The Make Utility 23.18 Summary Chapter 24 Network File System Concepts(NFS) 24.1 Introduction 24.2 Remote File Access Vs. Transfer 24.3 Operations On Remote Files 24.4 File Access Among Heterogeneous Computers 24.5 Stateless Servers 24.6 NFS And UNIX File Semantics 24.7 Review Of The UNIX File System 24.7.1 Basic Definitions 24.7.2 A Byte Sequence Without Record Boundaries 24.7.3 A File's Owner And Group Identifiers 24.7.4 Protection And Group Identifiers 24.7.5 The Open-Read-Write-Close Paradigm 24.7.6 Data Transfer 24.7.7 Permission To Search A Directory 24.7.8 Random Access 24.7.9 Seeking Beyond The End Of File 24.7.10 File Names And Paths 24.7.11 Semantics Of Write During Concurrent Access 24.7.12 File Names And Paths 24.7.13 Inode : Information Stored With A File 24.7.14 Stat Operation 24.7.15 The File Naming Mechanism 24.7.16 File System Mounts 24.7.17 File Name Resolution 24.7.18 Symbolic Links 24.8 Files Under NFS 24.9 NFS File Types 24.10 NFS File Modes 24.11 NFS File Attributes 24.12 NFS Client And Server 24.13 NFS Client Operation 24.14 NFS Client And UNIX Systems 24.15 NFS Mounts 24.16 File Handle 24.17 Handles Replace Path Names 24.18 File Positioning With A

Stateless Server 24.19 Operations On Directories 24.20 Reading A Directory Statelessly 24.21 Multiple Hierarchies In An NFS Server 24.22 The Mount Protocol 24.23 Transport Protocols For NFS 24.24 Summary Chapter 25 Network File System Protocols(NFS , Mount) 25.1 Introduction 25.2 Using RPC To Define A Protocol 25.3 Defining A Protocol With Data Structures And Procedures 25.4 NFS Constant , Type , And Data Declarations 25.4.1 NFS Constants 25.4.2 NFS Typedef Declarations 25.4.3 NFS Data Structures 25.5 NFS Procedures 25.6 Semantics Of NFS Operations 25.6.1 NFSPROC3_NULL(Procedure0) 25.6.2 NFSPROC3_GETATTR(Procedure1) 25.6.3 NFSPROC3_SETATTR(Procedure2) 25.6.4 NFSPROC3_LOOKUP(Procedure3) 25.6.5 NFSPROC3_ACCESS(Procedure4) 25.6.6 NFSPROC3_READLINK(Procedure5) 25.6.7 NFSPROC3_READ(Procedure6) 25.6.8 NFSPROC3_WRITE(Procedure7) 25.6.9 NFSPROC3_CREATE(Procedure8) 25.6.10 NFSPROC3_MKDIR(Procedure9) 25.6.11 NFSPROC3_SYMLINK(Procedure10) 25.6.12 NFSPROC3_MKNOD(Procedure11) 25.6.13 NFSPROC3_REMOVE(Procedure12) 25.6.14 NFSPROC3_RMDIR(Procedure13) 25.6.15 NFSPROC3_RENAME(Procedure14) 25.6.16 NFSPROC3_LINK(Procedure15) 25.6.17 NFSPROC3_READDIR(Procedure16) 25.6.18 NFSPROC3_READDIRPLUS(Procedure17) 25.6.19 NFSPROC3_FSSTAT(Procedure18) 25.6.20 NFSPROC3_FSINFO(Procedure19) 25.6.21 NFSPROC3_PATHCONF(Procedure20) 25.6.22 NFSPROC3_COMMIT(Procedure21) 25.7 The Mount Protocol 25.7.1 Mount Constant Definitions 25.7.2 Mount Type Definitions 25.7.3 Mount Data Structures 25.8 Procedures In The Mount Protocol 25.9 Semantics of Mount Operations 25.9.1 MOUNTPROC3_NULL(Procedure0) 25.9.2 MOUNTPROC3_MNT(Procedure1) 25.9.3 MOUNTPROC3_DUMP(Procedure2) 25.9.4 MOUNTPROC3_UMNT(Procedure3) 25.9.5 MOUNTPROC3_UMNTALL(Procedure4) 25.9.6 MOUNTPROC3_EXPORT(Procedure5) 25.10 NFS And Mount Authentication 25.11 File Locking 25.12 Changes In NFS Between Versions 3 And 4 25.13 Summary Chapter 26 A TELNET Client(Program Structure) 26.1 Introduction 26.2 Overview 26.2.1 The User's Terminal 26.2.2 Command And Control Information 26.2.3 Terminals , Windows , and Files 26.2.4 The Need For Concurrency 26.2.5 A Process Model For A TELNET Client 26.3 A TELNET Client Algorithm 26.4 Terminal I/O In Linux 26.4.1 Controlling A Device Driver 26.5 Establishing Terminal Modes 26.6 Global Variable Used For Stored State 26.7 Restoring Terminal Modes Before Exit 26.8 Client Suspension And Resumption 26.9 Finite State Machine Specification 26.10 Embedding Commands In A TELNET Data Stream 26.11 Option Negotiation 26.12 Request/Offer Symmetry 26.13 TELNET Character Definitions 26.14 A Finite State Machine For Data From The Server 26.15 Transitions Among States 26.16 A Finite State Machine Implementation 26.17 A Compact FSM Representation 26.18 Keeping The Compact Representation At Run-Time 26.19 Implementation Of A Compact Representation 26.20 Building An FSM Transition Matrix 26.21 The Socket Output Finite State Machine 26.22 Definitions For The Socket Output FSM 26.23 The Option Subnegotiation Finite State Machine 26.24 Definitions For The Option Subnegotiation FSM 26.25 FSM Initialization 26.26 Arguments For The TELNET Client 26.27 The Heart Of The TELNET Client 26.28 Implementation Of The Main FSM 26.29 Summary Chapter 27 A TENET Client(Implementation Details) 27.1 Introduction 27.2 The FSM Action Procedures 27.3 Recording The Type Of An Option Request 27.4 Performing No Operation 27.5 Responding To WILL/WONT For The Echo Option 27.6 Responding To WILL/WONT For Unsupported Options 27.7 Responding To WILL/WONT For The No Go-Ahead Option 27.8 Generating DO/DON'T For Binary Transmission 27.9 Responding To DO/DON'T For Unsupported Options 27.10 Responding To DO/DON'T For Transmit Binary Option 27.11 Responding To DO/DON'T For The Terminal Type Option 27.12 Option Subnegotiation 27.13 Sending Terminal Type Information 27.14 Terminating Subnegotiation 27.15 Sending A Character To The Server 27.16 Displaying Incoming Data On The User's Terminal 27.17 Using Termcap To Control The User's Terminal 27.18 Writing A Block Of Data To The Server 27.19 Interacting With The Client Process 27.20 Responding To Illegal Commands 27.21 Scripting To A File 27.22 Implementation Of Scripting 27.23 Initialization Of Scripting 27.24 Collecting Characters Of The Script File Name 27.25 Opening A Script File 27.26 Terminating Scripting 27.27 Printing Status Information 27.28 Summary Chapter 28 Streaming Audio And Video Transport(RTP Concept

And Design) 28.1 Introduction 28.2 Streaming Service 28.3 Real-Time Delivery 28.4 Protocol Compensation For Jitter 28.5 Retransmission , Loss , And Recovery 28.6 Real-Time Transport Protocol 28.7 Stream Translation And Jitter Buffers 28.9 RTP Control Protocol(RTCP) 28.10 Synchronizing Multiple Streams 28.11 RTP Transport And Many-To-Many Transmission 28.12 Sessions , Streams , Protocol Ports , And Demultiplexing 28.13 Basic Approaches To Encoding 28.14 Conceptual Organization Of RTP Software 28.15 Process/Thread Structure 28.16 Semantics Of The API 28.17 Jitter Buffer Design And Rebuffering 28.18 Event Handling 28.19 Playback Anomaly And Timestamp Complications 28.20 Size of An Example Real-Time Library 28.21 An Example MP3 Player 28.22 Summary Chapter 29 Streaming Audio And Video Transport(Example RTP Implementation 29.1 Introduction 29.2 An Integrated Implementation 29.3 Program Architecture 29.4 RTP Definitions 29.5 Manipulation Of Time Values 29.6 RTP Packet Queue Manipulation 29.7 RTP Packet Queue Manipulation 29.8 RTP Input Processing 29.9 Keeping Statistics For RTCP 29.10 RTP Initialization 29.11 RTCP Definitions 29.12 receiving RTCP Sender Reports 29.13 Generating RTCP Receiver Reports 29.14 RTCP Header Creation 29.15 RTCP Delay Computation 29.16 Generation Of An RTCP Bye Message 29.17 Size Of An Integrated Implementation 29.18 Summary Chapter 30 Practical Hints And Techniques For Linux Servers 30.1 Introduction 30.2 Operating In Background 30.3 Programming A Server To Operate In Background 30.4 Open Descriptors And Inheritance 30.5 Programming A Server To Close Inherited Descriptors 30.6 Signals From The Controlling TTY 30.7 Programming A Server To Change Its Controlling TTY 30.8 Moving To A Safe And Known Directory 30.9 Programming A Server To Change Directories 30.10 The Linux Umask 30.11 Programming A Server To Set Its Umask 30.12 Process Groups 30.13 Programming A Server To Set Its Process Group 30.14 Descriptors For Standard I/O 30.15 Programming A Server To Open Standard Descriptors 30.16 Mutual Exclusion For The Server 30.17 Programming A Server To Avoid Multiple Copies 30.18 Recording A Server's Process ID 30.19 Programming A Server To Record Its Process ID 30.20 Waiting For A Server To Wait For Each Child To Exit 30.22 Extraneous Signals 30.23 Programming A Server To Ignore Extraneous Signals 30.24 Using A System Log Facility 30.24.1 Generating Log Messages 30.24.2 The Advantage Of Indirection And Standard Error 30.24.3 Limitations Of I/O Redirection 30.24.4 A Client-Server Solution 30.24.5 The Syslog Mechanism 30.24.6 Syslog Message Classes 30.24.7 Syslog Facilities 30.24.8 Syslog Priority Levels 30.24.9 Using Syslog 30.24.10 An Example Syslog Configuration File 30.25 Summary Chapter 31 Deadlock And Starvation In Client-Server Systems 31.1 Introduction 31.2 Definition Of Deadlock 31.3 Difficulty Of Deadlock Detection 31.4 Deadlock Avoidance 31.5 Deadlock Between A Client And Server 31.6 Avoiding Deadlock In A Single Interaction 31.7 Starvation Among A Set Of Clients And A Server 31.8 Busy Connections And Starvation 31.9 Avoiding Blocking Operations 31.10 Processes , Connections , And Other Limits 31.11 Cycles Of Clients And Servers 31.12 Documenting Dependencies 31.13 Summary Appendix 1 System Calls And Library Routines Used With Sockets Appendix 2 Manipulation Of Linux File And Socket Descriptors Bibliography Index

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>