

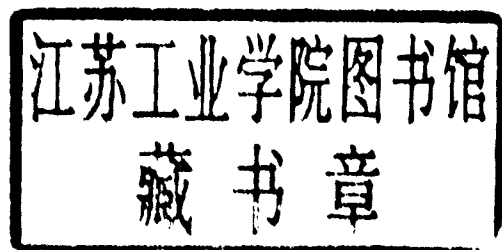


NETWORK COMPUTING SYSTEM TUTORIAL

TOM LYONS

Network Computing System Tutorial

Tom Lyons



Prentice Hall
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Editorial/production supervision: *Mary P. Rottino*

Cover design: *Lundgren Graphics*

Manufacturing buyer: *Kelly Behr/Susan Brunke*

Copyright © 1991 by Hewlett-Packard Company.



Published by Prentice-Hall, Inc.
A division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

The publisher offers discounts on this book when ordered
in bulk quantities. For more information, write:

Special Sales/College Marketing
Prentice-Hall, Inc.
College Technical and Reference Division
Englewood Cliffs, NJ 07632

UNIX is a registered trademark of AT&T in the USA and other countries.

MS-DOS® is a U.S. registered trademark of Microsoft Corporation.

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This document contains proprietary information which is protected by copyright. All rights reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company.

RESTRICTED RIGHTS LEGEND. Use, duplication, or disclosure by government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227.7013. Hewlett-Packard Co., 3000 Hanover St., Palo Alto, CA 94304.

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-617242-3

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada, Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Preface

The *Network Computing System Tutorial* explains how to use Version 1.5.1 of the Network Computing System (NCS) to create heterogeneous distributed computing applications. We've organized this tutorial as follows:

- | | |
|------------------|---|
| Chapter 1 | Discusses the basic concepts of using remote procedure calls in distributed computing and introduces the Network Computing Architecture (NCA) and the Network Computing System (NCS). |
| Chapter 2 | Shows how to make a remote procedure call. |
| Chapter 3 | Describes the basics of distributing an application. |
| Chapter 4 | Explores some of the details of distributing an application. |
| Chapter 5 | Catalogs NCS tools for distributed application development. |
| Chapter 6 | Demonstrates the design of a distributed application. |
| Chapter 7 | Demonstrates how to revise and extend a distributed application. |
| Chapter 8 | Demonstrates some recovery techniques. |
| Chapter 9 | Describes how to use the NCS object-oriented binding model. |

There is a glossary and an index at the end of the book.

About this Book

The first half of the book treats NCS as a conventional RPC system, a system that extends the procedure call mechanism over two machines connected by a network. Distributed applications are examined as extensions of conventional applications, with an emphasis on the features of NCS that make RPCs look like local procedure calls at the network interface. By the end of Chapter 3, nearly all of the ideas underlying NCS have been introduced. The features added in Chapter 4 improve the robustness and portability of applications, but

don't add anything fundamental to the discussion in the first three chapters. The first four chapters include nearly all of what normally characterizes commercial RPC systems today.

Chapter 5 is a survey of the tools NCS provides an application programmer. It reviews the Network Interface Definition Language (NIDL), the application library calls, and the NCS exception model. It also introduces NIDL features and NCS calls not encountered in previous examples, and describes some of the NCS support software. It thus forms a supplement to, but not a substitute for, the *Network Computing System Reference Manual*. By systematically introducing NIDL attributes, open arrays, open structures, and exception handling, Chapter 5 simplifies the presentation of advanced topics in the second half of the book.

The second part of the book emphasizes the differences between distributed computing and traditional single-process computing. NCS allows you to exploit those differences where they are useful to your application and hide them where they are not. It was designed to facilitate the building of reliable, large-scale, long-lived, portable, and extensible distributed systems. Whereas the first part of the book starts from the premise that you have a conventional local application that would benefit from being distributed, and goes on to show how to transparently “remote” an interface between two modules, the premise of the second part is that you want to create a new distributed application.

Chapter 6 begins with some suggestions for partitioning an application and designing good network interfaces. These suggestions are illustrated by creating a primitive **name server**, a network server program that maps names onto unique identifiers, and building an application around it. In Chapters 7 and 8, issues of maintenance, administration, and partial failure in distributed systems are identified in the process of extending the application of Chapter 6.

Chapter 9 introduces the object-oriented features of NCS: the use of UUIDs as invariant object identifiers, location transparency, and type-dependent binding of procedures. Object-orientation is central to the Network Computing Architecture (NCA) and NCS, but the casting of distributed computing in object-oriented terms can make both subjects harder to learn. That is why we've postponed the discussion of object-oriented distributed computing until the end of the book. Object-oriented solutions are a good fit to the problems of distributed computing, and we hope that Chapter 9 will help you take advantage of the object-oriented features of NCS. The *Network Computing System Reference Manual* includes more material on object-oriented programming with NCS.

Using this Book

We assume you have built applications in the C programming language. We don't assume you have any experience with networking software or remote procedure call (RPC) systems. The concepts of RPC and client/server computing as they apply to NCS are presented in Chapter 1. The networking basics you will need to understand NCS and use it effectively are presented in Chapter 2.

NCS can be used with a wide variety of programming languages, operating systems, and hardware architectures. However, it is impossible to cover in this book the full range of programming and operating environments supported by NCS, so we've had to make a few choices.

The sample programs in this book are written in C. The C language used in this book is basically that documented in the first edition of *The C Programming Language* by Kernighan and Ritchie (Prentice-Hall, 1978), with the addition of the `void` type specifier and structure assignment, but the programs are compatible with ANSI compilers as well.

We've tried to keep the examples portable. Unless otherwise noted, they should compile and run on any system that NCS runs on. Except where noted, the sample programs are demonstrated in the text by compiling and running them on an Apollo workstation under the Bourne shell but, aside from some differences noted in the text, the output should be virtually identical on any other UNIX* system.

If you are not using NCS on a UNIX system, you should know that the Bourne shell is a UNIX command line interpreter, which uses the dollar sign (\$) as a standard prompt. When a compiler command is too long to fit on a single line, we've divided the command onto multiple lines with an escaped newline (a backslash, \, followed by RETURN) to continue it on the next line. By default, the Bourne shell issues the secondary prompt > (or >> on some systems) when expecting another line to complete a command. Thus the following three lines are a single compiler command to link nine object modules into a single executable. The second dollar sign prompt (\$) indicates that the command has completed.

```
$ cc -o dbd12a dbd12a.o db12.o dba.o \  
> db1_sstub.o db2_sstub.o dba_sstub.o \  
> dba_cstub.o dba_cswtch.o util.o -lnck  
$
```

Pathnames and command names shown in the text are correct for most UNIX systems. The details of using NCS on various systems are described in the *Network Computing System Reference Manual*. You should also consult the *Release Document* for your NCS product to learn of any changes peculiar to your system.

Sample Programs

The examples in this book are available in machine-readable form from the publisher. The examples distribution contains all the software in this book, together with makefiles for building the programs on UNIX systems and some additional code needed to make the examples complete programs or applications that you can run and test. Even though this book does not contain listings of all the files in the distribution, all the software in the

* UNIX is a registered trademark of AT&T in the U.S.A. and other countries.

distribution that illuminates the subject of distributed programming with NCS is included in the figures of this book.

All the code in this book and in the examples distribution is in the public domain and may be available from other sources in addition to the publisher. If you have a copy of the sample programs, feel free to make it available to other readers.

Related Manuals

For more information on the Network Computing Architecture and Network Computing System, see the following documents, which we list with their Hewlett-Packard order numbers:

- *Network Computing Architecture* (010201-A01)

This book specifies the Network Computing Architecture in enough technical detail to write a new implementation that is compatible with NCS. It is mainly intended for programmers producing a new implementation of the architecture or porting an existing implementation, such as Hewlett-Packard's NCS, to a new platform. You do not need to read *Network Computing Architecture* in order to use this book. Architectural concepts from the NCA will be introduced and explained as needed in the text.

- *Network Computing System Reference Manual* (D-10200-C)

This book is a comprehensive programmer's reference for NCS. You will want to consult it whenever you have a question that is not answered in this manual, or want to learn about features not used for the examples in this book. We will often refer to the reference when there is more to a subject than can be discussed here.

- *Managing NCS Software* (D-11895-C)

This book explains how to set up and administer NCS software, including the Global and Local Location Broker Daemons. It is recommended to administrators of networks running software based on NCS. Programmers should find the information in the Release Document accompanying the NCS product sufficient for setting up the system required to compile and run the examples in this book.

The Hewlett-Packard order number for the *Network Computing System Tutorial* is (D-18355-B).

The *Release Document* for each NCS product from Hewlett-Packard contains installation procedures, descriptions of new or changed features, and lists of known and fixed bugs.

Typographic Conventions

Unless otherwise noted in the text, this manual uses the following conventions.

literal values	Bold words or characters in formats and command descriptions represent commands or keywords that you must use literally. Pathnames are also in bold. Bold words in text indicate the first use of a new term. In interactive examples, characters that you type appear in bold.
<i>user-supplied values</i>	Italic words or characters in formats and command descriptions represent values that you must supply.
output/source code	Information that the system displays appears in this typeface. Examples of source code also appear in this typeface.
^C	This indicates a program interrupt generated from the keyboard. On most UNIX systems, you can generate an interrupt by typing C while holding down the CTRL key.
⋮	A vertical ellipsis means that irrelevant parts of a figure or example have been omitted.
———— ☐ ————	This symbol indicates the end of a chapter or part of a manual.

Acknowledgements

I want to thank Mike Kong for the *Network Computing System Reference Manual*, and his early guidance in the writing of this book. Special thanks go to Kevin Ackley of Course Six, Inc. for his close reading of several drafts, and for convincing me to write Chapter 2. My thanks also to Nat Mishkin, Liza Martin, and Dale Labossiere of the NCS team at Hewlett-Packard for reviewing the text, correcting errors of fact and emphasis, and answering numerous questions. I also want to express deep appreciation to Lisa Zahn and Richard Curtis for recognizing the need for a tutorial, and allowing me the time to write one. My sincerest thanks to Marilyn Kotwal, the editor, for her unflagging attention to language and detail. And finally, thanks to the NCS team past and present for writing such useful and elegant software.

Tom Lyons



Contents

Preface	xiii
About this Book	xiii
Using this Book	xv
Sample Programs	xv
Related Manuals	xvi
Typographic Conventions	xvii
Acknowledgements	xviii
 Chapter 1 The Network Computing Architecture and Network Computing System	 1
1.1 The Distributed Computing Environment	1
1.2 NCA	2
1.2.1 Remote Procedure Calls	3
1.2.2 The Network Interface Definition Language	5
1.2.3 The Network Data Representation	6
1.2.4 Clients and Servers	7
1.2.5 Brokers	7
1.3 A Summary of NCA	8
1.4 NCS	9
1.4.1 What You Need to Get Started	10
1.4.2 NCS and NCA	10
 Chapter 2 Calling A Remote Procedure	 13
2.1 The <code>rrpc_\$inq_stats</code> Call	13
2.1.1 RPC Handles	14
2.1.2 UUIDs and <code>uuid_\$nil</code>	15
2.1.3 A Digression on Sockets	16
2.1.4 How NCS Uses Socket Addresses	17
2.1.5 Handles and Binding	17
2.2 Compiling the Program	18
2.2.1 Dealing with <code>\$s</code>	18
2.2.2 Using the NCS C Preprocessor	19

2.3	Running the Program	19
2.4	What's Happening?	20
2.5	Making Repeated Calls to a Single Server	23
2.6	How NCS Represents Socket Addresses	24
2.7	Identifying a Server	25
2.8	Specifying a Server	28
2.9	Other Servers	30
Chapter 3	A Simple Distributed Application	31
3.1	A Conventional Application	31
3.2	A Distributed Application	34
3.3	The Network Interface Definition File	34
3.3.1	Attributes	35
3.3.2	The NIDL Compiler	36
3.3.3	Compiling the Stubs	37
3.3.4	How Stubs Work	39
3.3.5	Stub Code	40
3.4	Binding with a Generic Handle	40
3.4.1	The <code>_bind</code> Handle Binding Function	43
3.4.2	The <code>_unbind</code> Handle Releasing Function	43
3.4.3	Caching RPC Handles	43
3.4.4	Compiling the Handle Binding Code	44
3.5	Specifying Pathnames in Distributed Applications	44
3.6	The Server Code	45
3.6.1	Registering with NCK	47
3.6.1.1	The Interface Specification	47
3.6.1.2	The Server EPV	47
3.6.1.3	The Manager EPV	48
3.6.1.4	The <code>rpc_\$register_mgr</code> Call	48
3.6.1.5	The <code>lb_\$register</code> Call	48
3.6.2	Servicing RPCs	49
3.6.3	Catching Signals and Faults	50
3.6.4	Compiling the Server Code	51
3.7	Building the Distributed Application	51
3.8	Running the Distributed Application	54
Chapter 4	Transmittable Types, Error Reporting and Address Family Independence ...	57
4.1	Data Types in NIDL and C	57
4.1.1	Pointers, Arrays, and Strings	60
4.1.2	Stubs and Prototype Declarations	63
4.2	Status Checking	64
4.2.1	A Simple Completion Status Checker	64
4.2.2	Adding Status Checking to the Application	65
4.2.3	Rebuilding and Running the Application	70

4.3	Address Family Independence	72
4.3.1	Registering in Multiple Address Families	73
4.3.2	Binding in Multiple Address Families	77
Chapter 5	A Tour of NCS	81
5.1	NIDL	81
5.1.1	The Format of a NIDL File	82
5.1.2	The Interface Identifier and Interface Attribute List	83
5.1.3	Import Declarations	85
5.1.4	Constant Declarations	86
5.1.5	Type Declarations	87
5.1.5.1	NIDL Scalar Type Specifiers	88
5.1.5.2	Constructed Types	92
5.1.5.3	Arrays, <code>last_is</code> and <code>max_is</code>	92
5.1.5.4	Strings	99
5.1.5.5	Structures	99
5.1.5.6	Unions	100
5.1.5.7	Pointers	102
5.1.6	Operation Declarations	103
5.1.6.1	Operation Attributes	103
5.1.6.2	Parameter Attributes	105
5.2	Using <code>transmit_as</code> to Marshal an Array of Pointers	106
5.3	Invoking the NIDL Compiler	115
5.4	A Few Last Words on NIDL and <code>nidl</code>	116
5.5	The NCK Library	118
5.5.1	The <code>lb_\$</code> Calls	120
5.5.2	The <code>rpc_\$</code> and <code>rrpc_\$</code> Calls	121
5.5.3	The <code>socket_\$</code> Calls	123
5.5.4	The <code>uuid_\$</code> Calls	126
5.6	Error and Fault Handling Calls	128
5.6.1	Establishing Cleanup Handlers	129
5.6.2	Leaving Cleanup Handlers	130
5.6.3	Releasing Cleanup Handlers	132
5.6.4	Enabling Faults on Leaving a Cleanup Handler	134
5.6.5	Inhibiting Asynchronous Faults	134
5.6.6	The Volatile Macro	135
5.6.7	More on <code>status_\$t</code>	136
5.6.8	Detecting Communication Faults	137
5.6.9	Using <code>comm_status</code> to Trap Communication Faults	138
5.6.10	PPFM and Signals	139
5.6.11	Exception Handling in Distributed Applications	140
5.7	Daemons and Support Tools	141
5.7.1	The Location Broker	141
5.7.1.1	The Local Location Broker Daemon	141
5.7.1.2	The Global Location Broker Daemon	141
5.7.1.3	The LB Administration Tools	142
5.7.2	The UUID Generator	143
5.7.3	The Status Code Interpreter	144

Chapter 6	Designing a Distributed Application	145
6.1	Traditional Programming with Procedure Calls	145
6.2	Programming with RPCs	146
6.2.1	Narrow Interfaces	146
6.2.2	Dynamic Binding	147
6.2.3	Designing for NCS	147
6.3	Using Encapsulation	148
6.4	Designing a Good Interface	148
6.5	A Simple Database Manager	152
6.5.1	A Digression on Defining the Manager EPV	159
6.5.2	Rapid Prototyping	161
6.6	The Complete Database Server	161
6.6.1	Building the Database Daemon	168
6.6.2	Running the Database Daemon	169
6.7	Using the Database Daemon	170
6.7.1	Modifying the Message Server Program	171
6.7.2	Building and Running the User Daemon Program	180
6.7.3	Modifying the Message Client Program	182
6.7.4	Building the New Message Client Program	185
6.7.5	Putting all the Pieces Together	186
6.8	Using make to Maintain Distributed Applications	188
6.8.1	NIDL File Dependency Rules	190
6.8.2	Building the Application with make	191
Chapter 7	Developing a Distributed Application	193
7.1	A Digression on Address Family Support	193
7.2	Creating a New Interface Version	194
7.2.1	A Manager for Version 2 of the db Interface	196
7.2.2	A New Database Daemon	202
7.2.3	Updating the User Message Daemon	208
7.2.4	Updating the Message Sending Program	218
7.2.5	A Makefile for Version 2 of the Application	221
7.2.6	Running Version 2 of the Application	222
7.3	Supporting Obsolete Clients	226
7.3.1	Implementing Multiple Interface Versions in a Manager	227
7.3.2	Exporting Multiple Versions of an Interface from a Server	228
7.3.3	Building and Running the New Database Daemon	236
7.4	Extending an Interface	238
7.5	Adding a New Interface to a Server	242
Chapter 8	Recovery Techniques	253
8.1	Detecting Multiple Instances of a Server	253
8.2	Detecting Server Crashes	255
8.2.1	Using rrpc_\$are_you_there	256
8.2.2	Long and Short RPC Timeouts	259
8.2.3	Calling a dba Operation from the dba Server	262

8.2.4	Using the Client EPV	264
8.3	Adding Persistence to a Server	267
8.3.1	Save and Restore Functions	268
8.4	Shutting Down a Server	270
Chapter 9	Object-Oriented Distributed Computing	273
9.1	Using the Object Identifier	273
9.1.1	Binding a Handle on an Object	275
9.1.2	Registering an Object with NCS	276
9.1.3	Registering a User as an Object	278
9.2	Locating Objects	281
9.2.1	The Structure of an LB Entry	281
9.2.2	The LB Lookup Calls	282
9.3	The NCS Object Model	283
9.3.1	Objects and Types	283
9.3.2	Types and Interfaces	284
9.3.3	Object-Oriented Binding	291
9.4	Object Orientation in Distributed Systems	305
9.4.1	Re-use with Encapsulation	306
9.4.2	Flexible Binding	306
9.4.3	Object-Oriented Programming in the Large	306
	Glossary	309
	Index	319

Figures

1-1.	The Heterogeneous Distributed Computing Environment	2
1-2.	A Local Procedure Call	3
1-3.	Transfer of Control in a Local Procedure Call	4
1-4.	Transfer of Data in a Remote Procedure Call	4
1-5.	Transfer of Control in a Remote Procedure Call	5
1-6.	The Client and Server of an Interface	7
1-7.	Client and Server Relationships	7
1-8.	Structure of the Network Computing Architecture	8
2-1.	Making a Remote Procedure Call: rpcall1.c	15
2-2.	Making a Series of RPCs to the Same Server: rpcall2.c	23
2-3.	Identifying a Server: rpcall3.c	26
2-4.	Specifying a Server: rpcall4.c	29
3-1.	The client.c File	32
3-2.	The manager.c File	32
3-3.	The ms.h File	32
3-4.	A Local Application	33
3-5.	A NIDL Specification: ms.idl	34
3-6.	Translating an Interface Definition into Stub and Header Files	38
3-7.	Stubs Make RPCs Appear Local to the Application Code	39
3-8.	Handle Binding and Unbinding: bind.c	42
3-9.	Registration and Unregistration Code: server.c	46
3-10.	Cleanup Handler Code: in server.c	50
3-11.	The Completed Distributed Application	53
4-1.	A Simple Network Time of Day Interface	58
4-2.	Client Calling www_current_time	59
4-3.	A NIDL Specification: ms.idl	61
4-4.	How a Parameter Declaration Affects String Marshaling	62
4-5.	Declaration of ms_show Produced by NIDL Compiler: in ms.h	63
4-6.	A Function for Testing Completion Statuses: util.c	65
4-7.	Completion Status Checking: bind.c	66
4-8.	Completion Status Checking: server.c	68
4-9.	Inquiring about Supported Address Families: in server.c	74
4-10.	Registering a Server on Multiple Address Families: in server.c	75
4-11.	Unregistering a Server in Multiple Address Families: in server.c	76
4-12.	Binding in Multiple Address Families: in bind.c	78
4-13.	The ms_string_t_unbind Function: in bind.c	79
5-1.	Casting Constants when Passed to NCS Calls	91
5-2.	Marshaling and Unmarshaling an Open Array with max_is and last_is	95
5-3.	An Argument Vector of Type char *argv[]	107
5-4.	A Network Interface for Remote Command-Line Execution: rex.idl	108
5-5.	Translating an Argument Vector into an Open Structure: in rex_xmit.c	111
5-6.	The Array args in the Transmittable Structure of Type rex_args_t	112

5-7.	Recovering an Argument Vector from an Open Structure: in <code>rex_xmit.c</code>	113
5-8.	Data Structure Recovered by <code>rex_argv_t_from_xmit_rep</code>	114
5-9.	Functions To Free Storage Allocated for Translation: in <code>rex_xmit.c</code>	114
5-10.	Layout of a <code>status_t</code> Value	136
6-1.	Message System Database Interface: <code>db1.idl</code>	150
6-2.	Definition of Types and Constants for the <code>db</code> Interface: <code>db0.idl</code>	151
6-3.	Message Client and Server as Clients of the Database Server	152
6-4.	Include Files and Global Data Structures: in <code>db1.c</code>	153
6-5.	Private <code>set_address</code> and <code>new_entry</code> Functions: in <code>db1.c</code>	154
6-6.	Public <code>db_register</code> Operation: in <code>db1.c</code>	156
6-7.	Public <code>db_unregister</code> Operation: in <code>db1.c</code>	157
6-8.	Public <code>db_lookup</code> Operation: in <code>db1.c</code>	158
6-9.	The Manager EPV for the <code>db_v1</code> Interface, <code>db_v1_manager_epv</code> : in <code>db1.c</code>	158
6-10.	The <code>main</code> Function: in <code>dbd1.c</code>	162
6-11.	Global Declarations: in <code>dbd1.c</code>	163
6-12.	The Initialize Function in <code>dbd1.c</code>	164
6-13.	The <code>get_socket</code> Function: in <code>dbd1.c</code>	165
6-14.	Registration Code: in <code>dbd1.c</code>	166
6-15.	The Unregistration Code: in <code>dbd1.c</code>	167
6-16.	New Utility Functions: <code>util.c</code>	168
6-17.	Declarations of Utility Functions: <code>util.h</code>	169
6-18.	Global Declarations: in <code>userd1.c</code>	171
6-19.	The Initialize and <code>get_socket</code> Functions: in <code>userd1.c</code>	172
6-20.	The <code>register_name</code> Function: in <code>userd1.c</code>	173
6-21.	The <code>bind_db_handle</code> Function: in <code>userd1.c</code>	174
6-22.	The Register Function: in <code>userd1.c</code>	175
6-23.	The Unregistration Code: in <code>userd1.c</code>	177
6-24.	The <code>main</code> Function: in <code>userd1.c</code>	179
6-25.	The <code>ms</code> User Manager: <code>ms_user.c</code>	180
6-26.	Binding a Handle on a User Name: in <code>bind1.c</code>	182
6-27.	The <code>ms_string_t_unbind</code> Function: in <code>bind1.c</code>	183
6-28.	The <code>bind_db_handle</code> Function: in <code>bind1.c</code>	184
6-29.	The <code>ms</code> Client Code for Sending Messages: <code>sendmsg.c</code>	185
6-30.	Makefile for the New Message System	189
7-1.	New Version of <code>db</code> Interface: <code>db2.idl</code>	195
7-2.	New Type and Constant Declarations for <code>db</code> Interface: <code>db0.idl</code>	196
7-3.	Global Declarations and Data Structures: in <code>db2.c</code>	197
7-4.	The <code>set_address</code> Function: in <code>db2.c</code>	198
7-5.	The <code>new_entry</code> Function and <code>db_register</code> Operation: in <code>db2.c</code>	199
7-6.	The <code>db_unregister</code> Operation: in <code>db2.c</code>	200
7-7.	The <code>db_lookup_name</code> Operation: in <code>db2.c</code>	201
7-8.	Definition of the Version 2 Manager EPV: in <code>db2.c</code>	202
7-9.	A Server for Version 2 of the <code>db</code> Interface: in <code>dbd2.c</code>	203
7-10.	The <code>get_sockets</code> Function: in <code>dbd2.c</code>	204
7-11.	Server Registration Code: in <code>dbd2.c</code>	205

7-12.	Server Unregistration Code: in dbd2.c	206
7-13.	The main Function: in dbd2.c	207
7-14.	Global Declarations and Initialize Function: in userd2.c	209
7-15.	The get_sockets Function: in userd2.c	210
7-16.	The bind_db_handles Function: in userd2.c	211
7-17.	The register_name Function: in userd2.c	212
7-18.	The Register Function: in userd2.c	214
7-19.	The unregister_uuid Function: in userd2.c	215
7-20.	The Unregister Function: in userd2.c	216
7-21.	Function main : in userd2.c	217
7-22.	The bind_db_handle Function: in bind2.c	219
7-23.	The ms_string_t_bind and ms_string_t_unbind Functions: in bind2.c	220
7-24.	A Makefile for Version 2 of the Application	221
7-25.	Defining Multiple Manager EPVs and a db_lookup Operation: db12.c	228
7-26.	Multiple Manager EPVs and Other Global Declarations: in dbd12.c	229
7-27.	The Initialize Function: in dbd12.c	230
7-28.	The get_sockets Function: in dbd12.c	231
7-29.	The Register Function: in dbd12.c	233
7-30.	The Unregister Function: in dbd12.c	234
7-31.	The main Function: in dbd12.c	235
7-32.	Makefile for Building the Multiple Version Database Daemon dbd12	237
7-33.	New db_lookup_uuid Operation Added to db_v2 interface: db2.idl	239
7-34.	Implementation of db_lookup_uuid operation: in db12.c	240
7-35.	New Database Interface: dba.idl	243
7-36.	Manager Module for dba Interface: dba.c	244
7-37.	Header Files and Manager EPV Declarations in dbd12a.c	245
7-38.	Global Declarations in dbd12a.c	246
7-39.	The Initialize Function in dbd12a.c	247
7-40.	The get_sockets Function in dbd12a.c	248
7-41.	The Register Function: in dbd12a.c	249
7-42.	The Unregister Function: in dbd12a.c	250
7-43.	The main Function: in dbd12a.c	251
8-1.	A Function to Check for Multiple Database Daemons: in dbd12a.c	255
8-2.	Testing Whether a Server is Still Responding to RPCs: in dbd12a.c	257
8-3.	A check_for_daemons Function That Calls server_responding : in dbd12a.c ...	258
8-4.	Using a Short Timeout in the server_responding Function: in dbd12a.c	261
8-5.	A server_responding Function Specialized for the dba Interface: in dbd12a.c ..	262
8-6.	Calling a Remote Operation Directly Through the Client EPV: in dbd12a.c	266
8-7.	Function to Save the Contents of the Name Database: in dbio.c	268
8-8.	Function to Restore the Contents of the Name Database: in dbio.c	269
8-9.	A New main Function to Save and Restore Database Contents: in dbd12a.c	270
8-10.	New main Function to Enable Remote Shutdowns in dbd12a.c	271
9-1.	Specifying an Object UUID in the ms_string_t_bind Function: in bind.c	275
9-2.	Defining the User Type UUID: id_defs.h	277
9-3.	Defining the User Type UUID: in userd.c	278