Introduction 1

Introduction to This Guide

This guide provides information to developers on the use of the STREAMS mechanism at user and kernel levels. STREAMS was developed to augment the character input/output mechanism and to support development of communication services.

STREAMS provides developers with integral functions, a set of utility routines, and facilities that speed software design and implementation.

Audience

The guide is intended for network and systems programmers who use the STREAMS mechanism at user and kernel levels for UNIX system communication services.

Readers of the guide are expected to possess prior knowledge of the UNIX system, programming, networking, and data communication. It is also assumed that the reader is familiar with the book *Writing Device Drivers*.

Organization

This guide has several chapters, each discussing a unique topic. Chapters 2, 3, and 4 have introductory information and can be ignored by those readers already familiar with STREAMS concepts and facilities.

- Chapter 1, “Introduction" describes the organization and purpose of the guide. It also defines an intended audience and an expected background of the users of the guide.
- Chapter 2, “Overview of STREAMS" presents an overview and the benefits of STREAMS.
- Chapter 3, “STREAMS Mechanism" describes the basic operations for constructing, using, and dismantling Streams. These operations are performed using `open`(2), `close`(2), `read`(2), `write`(2), and `ioctl`(2).
- Chapter 4, “STREAMS Processing Routines" gives an overview of the STREAMS `put` and `service` routines.
- Chapter 5, “Messages" discusses STREAMS messages, their structure, linkage, queuing, and interfacing with other STREAMS components.
- Chapter 6, “Polling and Signaling" describes how STREAMS allows user processes to monitor, control, and poll Streams to allow an effective use of system resources.
- Chapter 7, “Overview of Modules and Drivers" describes the STREAMS module and driver environment, `ioctls`, routines, declarations, flush handling, and driver–kernel interface. It also provides general design guidelines for modules and drivers.
- Chapter 8, “Modules" provides information on module construction and function.
- Chapter 9, “Drivers" discusses STREAMS drivers, elements of driver flow control, flush handling, cloning, and processing.
- Chapter 10, “Multiplexing" describes the STREAMS multiplexing facility.
- Chapter 11, “STREAMS-Based Pipes and FIFOS" provides information on creating MS-based pipes and FIFOs and unique connections.
- Chapter 12, “STREAMS-Based Terminal Subsystem" discusses STREAMS-based terminal and pseudo-terminal subsystems.
- Chapter 13, “Multi-Threaded STREAMS" describes the multithreaded environment and what needs to be done to make your module or driver MT- safe.
- Appendix A, “STREAMS Data Structures" summarizes data structures commonly used by STREAMS modules and drivers.
- Appendix B, “Message Types" describes STREAMS messages and their use.

- Appendix C, “STREAMS Utilities" describes STREAMS utility routines and their use.
- Appendix D, “Debugging" provides debugging aids for developers.
- Appendix E, “Configuration" describes how modules and drivers are configured into the UNIX system, tunable parameters, and the autopush facility.
- Appendix F, “Manual Pages" lists STREAMS-related manual pages.
- “Glossary" defines terms unique to STREAMS.

Code Examples

All code examples used in this book conform to ANSI C specifications.

Conventions Used

Throughout this guide, the word “STREAMS” refers to the mechanism and the word “Stream” refers to the path between a user application and a driver. In connection with STREAMS-based pipes, “Stream” refers to the data transfer path in the kernel between the kernel and one or more user processes.

Examples are given to highlight the most important and common capabilities of STREAMS. They are not exhaustive and, for simplicity, reference fictional drivers and modules. Where possible, examples will be runnable code.

Command names, C code, UNIX code, system calls, STREAMS utility routines, header files, data structures, declarations, short examples, filenames, and path names are printed in `listing (constant width) font`.

User input is in `listing font` when by itself, or **`bold listing font`** when used in combination with computer output.

```
Screens are used to simulate what a user will see on a
video display screen or to show program source code.

Data structure contents and formats are also shown in screens.
```

Items being emphasized, variable names, and parameters are printed in *italics*.

Warning – The warning sign is used to show possible damage to data, system, application, or person.

Caution – The caution sign is used to show possible harm or damage to a system, an application, a process, or a piece of hardware.

Note – Notes are used to emphasize points of interest, to present parenthetical information, and to cite references to other documents and commands.

Other Documentation

Though the STREAMS Programming Guide is a principal tool to aid in developing STREAMS applications, readers are encouraged to obtain more information on both system calls and utilities used by STREAMS from the manual pages. See Appendix F, "Manual Pages"" for a complete list of the available manual pages. The actual manual pages are in the SunOS *man Pages(2): System Calls* and the *man Pages(3): Library Routines*.

For more information on driver related issues, such as autoconfiguration, see *Writing Device Drivers*.

For a complete list of books about SunOS/SVR4, see the Solaris Roadmap to Documentation.