Linux Fast-STREAMS Installation and Reference Manual

Version 0.9.2 Edition 4
Updated 2008-10-31
Package streams-0.9.2.4

Brian Bidulock <bidulock@openss7.org> for
The OpenSS7 Project <http://www.openss7.org/>

Published by OpenSS7 Corporation
1469 Jefferys Crescent
Edmonton, Alberta T6L 6T1
Canada

This is texinfo edition 4 of the Linux Fast-STREAMS manual, and is consistent with streams 0.9.2. This manual was developed under the OpenSS7 Project and was funded in part by OpenSS7 Corporation.

Short Contents

Table of Contents

Preface

Notice

This package is released and distributed under the *AGPL* (see [GNU Affero General Public License], page 144). Please note, however, that there are different licensing terms for the manual pages and some of the documentation (derived from OpenGroup[1] publications and other sources). Consult the permission notices contained in the documentation for more information.

This manual is released under the *FDL* (see [GNU Free Documentation License], page 170) with no sections invariant.

Abstract

This manual provides a *Installation and Reference Manual* for *Linux Fast-STREAMS*.

Objective

The objective of this manual is to provide a guide for the *STREAMS* programmer when developing *STREAMS* modules, drivers and application programs for *Linux Fast-STREAMS*.

This guide provides information to developers on the use of the *STREAMS* mechanism at user and kernel levels.

STREAMS was incorporated in UNIX System V Release 3 to augment the character input/output (I/O) mechanism and to support development of communication services.

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

Intent

The intent of this manual is to act as an introductory guide to the *STREAMS* programmer. It is intended to be read alone and is not intended to replace or supplement the *Linux Fast-STREAMS* manual pages. For a reference for writing code, the manual pages (see `STREAMS(9)`) provide a better reference to the programmer. Although this describes the features of the *Linux Fast-STREAMS* package, OpenSS7 Corporation is under no obligation to provide any software, system or feature listed herein.

Audience

This manual is intended for a highly technical audience. The reader should already be familiar with *Linux* kernel programming, the *Linux* file system, character devices, driver input and output, interrupts, software interrupt handling, scheduling, process contexts, multiprocessor locks, etc.

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

Readers of the guide are expected to possess prior knowledge of the *Linux* and *UNIX* system, programming, networking, and data communication.

[1] Formerly X/Open and UNIX International.

Revisions

Take care that you are working with a current version of this manual: you will not be notified of updates. To ensure that you are working with a current version, contact the Author, or check The OpenSS7 Project website for a current version.

A current version of this manual is normally distributed with the *Linux Fast-STREAMS* package.

Version Control

```
STREAMS.texi,v
Revision 0.9.2.45  2008-09-20 11:04:35  brian
- added package patchlevel

Revision 0.9.2.44  2008-08-03 06:03:36  brian
- protected agains texinfo commands in log entries

Revision 0.9.2.43  2008/07/27 08:49:17  brian
- no invariant sections, more libtool ignores

Revision 0.9.2.42  2008-04-28 12:54:00  brian
- update file headers for release

Revision 0.9.2.41  2008-04-25 11:50:49  brian
- updates to AGPLv3

Revision 0.9.2.40  2007/12/15 20:19:44  brian
- updates

Revision 0.9.2.39  2007/11/06 10:27:04  brian
- miscellaneous corrections

Revision 0.9.2.38  2007/08/12 06:44:32  brian
- updated licenses in manuals

Revision 0.9.2.37  2007/03/17 08:31:56  brian
- corrected formatting problems

Revision 0.9.2.36  2007/02/28 06:30:55  brian
- updates and corrections, #ifdef instead of #if

Revision 0.9.2.35  2007/01/02 16:32:03  brian
- updates for release, disable streams-bcm by default

Revision 0.9.2.34  2006/12/31 13:26:37  brian
- documentation updates for release

Revision 0.9.2.33  2006/09/18 01:06:57  brian
- updated manuals and release texi docs

Revision 0.9.2.32  2006/08/28 10:47:05  brian
- correction

Revision 0.9.2.31  2006/08/28 10:32:54  brian
- updated references

Revision 0.9.2.30  2006/08/27 12:26:58  brian
- finalizing auto release files
```

```
Revision 0.9.2.29  2006/08/26 18:31:44  brian
- handle long urls

Revision 0.9.2.28  2006/08/26 09:18:27  brian
- better release file generation

Revision 0.9.2.27  2006/08/23 11:00:41  brian
- added preface, corrections and updates for release

Revision 0.9.2.26  2006/08/22 12:36:49  brian
- udpates to documentation, tweaks to Stream head

Revision 0.9.2.25  2006/03/22 10:02:04  brian
- added makefile target index

Revision 0.9.2.24  2006/03/03 10:57:11  brian
- 32-bit compatibility support, updates for release

Revision 0.9.2.23  2005/09/15 13:03:08  brian
- added new graphics and updates

Revision 0.9.2.22  2005/07/08 13:16:11  brian
- updates to documentation

Revision 0.9.2.21  2005/06/24 13:38:59  brian
- added troubleshooting section to manuals

Revision 0.9.2.20  2005/05/14 08:34:34  brian
- copyright header correction

Revision 0.9.2.19  2005/04/15 00:58:31  brian
- working up documentation

Revision 0.9.2.18  2005/04/14 08:06:09  brian
- added figures

Revision 0.9.2.17  2005/04/12 09:28:59  brian
- corrections

Revision 0.9.2.16  2005/04/11 20:48:41  brian
- documentation updates and corrections

Revision 0.9.2.15  2005/03/15 12:06:58  brian
- Updated texinfo documentation.

Revision 0.9.2.14  2005/03/15 00:56:42  brian
- Updated version numbering in texinfo files.

Revision 0.9.2.13  2005/03/15 00:51:34  brian
- Updated version numbering in texinfo files.

Revision 0.9.2.12  2005/02/17 22:57:34  brian
- Some cross-reference corrections.

Revision 0.9.2.11  2005/02/17 11:34:53  brian
- Corrected some more texi problems.
```

```
Revision 0.9.2.10  2005/01/24 11:57:57  brian
- Updated texinfo headers.

Revision 0.9.2.9  2004/12/19 15:15:02  brian
- Corrected include position.

Revision 0.9.2.8  2004/12/17 04:02:46  brian
- Improving spec files.

Revision 0.9.2.7  2004/11/06 10:24:35  brian
- Updated documentation.

Revision 0.9.2.6  2004/08/22 07:28:53  brian
- Converted to shared common files.

Revision 0.9.2.5  2004/08/22 06:17:50  brian
- Checkin on new working branch.

Revision 0.9.2.4  2004/08/15 19:59:29  brian
- Build system updates.

Revision 0.9.2.3  2004/05/29 08:28:01  brian
- Working up stable release.

Revision 0.9.2.2  2004/03/15 07:59:39  brian
- Working up manual pages.

Revision 0.9.2.1  2004/03/13 05:46:34  brian
- Working up more documentation.
```

ISO 9000 Compliance

Only the TEX, texinfo, or roff source for this manual is controlled. An opaque (printed, postscript or portable document format) version of this manual is an **UNCONTROLLED VERSION**.

Disclaimer

OpenSS7 Corporation disclaims all warranties with regard to this documentation including all implied warranties of merchantability, fitness for a particular purpose, non-infringement, or title; that the contents of the manual are suitable for any purpose, or that the implementation of such contents will not infringe on any third party patents, copyrights, trademarks or other rights. In no event shall *OpenSS7 Corporation* be liable for any direct, indirect, special or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with any use of this manual or the performance or implementation of the contents thereof.

OpenSS7 Corporation reserves the right to revise this software and documentation for any reason, including but not limited to, conformity with standards promulgated by various agencies, utilization of advances in the state of the technical arts, or the reflection of changes in the design of any techniques, or procedures embodied, described, or referred to herein. *OpenSS7 Corporation* is under no obligation to provide any feature listed herein.

U.S. Government Restricted Rights

If you are licensing this Software on behalf of the U.S. Government ("Government"), the following provisions apply to you. If the Software is supplied by the Department of Defense ("DoD"), it is classified as "Commercial Computer Software" under paragraph 252.227-7014 of the DoD Supplement to the Federal Acquisition Regulations ("DFARS") (or any successor regulations) and the Government is acquiring only the license rights granted herein (the license rights customarily provided to non-Government users). If the Software is supplied to any unit or agency of the Government other than DoD, it is classified as "Restricted Computer Software" and the Government's rights in the Software are defined in paragraph 52.227-19 of the Federal Acquisition Regulations ("FAR") (or any successor regulations) or, in the cases of NASA, in paragraph 18.52.227-86 of the NASA Supplement to the FAR (or any successor regulations).

Acknowledgements

As with most open source projects, this project would not have been possible without the valiant efforts and productive software of the *Free Software Foundation* and the *Linux Kernel Community*.

Sponsors

Funding for completion of the *OpenSS7 Linux Fast-STREAMS* package was provided in part by:

- OpenSS7 Corporation

Additional funding for The OpenSS7 Project was provided by:

- OpenSS7 Corporation
- Lockheed Martin Co.
- Motorola
- HOB International
- Comverse Ltd.
- Sonus Networks Inc.
- France Telecom
- SS8 Networks Inc.
- Nortel Networks
- Verisign
- eServGlobal (NZ) Pty Ltd.
- NetCentrex S. A.
- SysMaster Corporation
- GeoLink SA
- AirNet Communications
- TECORE
- Tumsan Oy
- Vodare Ltd.
- Excel Telecommunications

Contributors

The primary contributor to the *OpenSS7 Linux Fast-STREAMS* package is Brian F. G. Bidulock. The following is a list of significant contributors to The OpenSS7 Project:

- Per Berquist
- John Boyd
- Chuck Winters
- Peter Courtney
- Tom Chandler
- Gurol Ackman
- Kutluk Testicioglu
- John Wenker
- Others

Authors

The authors of the *OpenSS7 Linux Fast-STREAMS* package include:

- Brian Bidulock

See [Index of Authors], page 186, for a complete listing and cross-index of authors to sections of this manual.

Maintainer

The maintainer of the *OpenSS7 Linux Fast-STREAMS* package is:

- Brian Bidulock

Please send bug reports to `bugs@openss7.org` using the '`send-pr`' script included in the package, only after reading the '`BUGS`' file in the release, or See Section 9.2 [Problem Reports], page 137.

Web Resources

The OpenSS7 Project provides a website dedicated to the software packages released by the OpenSS7 Project.

Bug Reports

Please send bug reports to `bugs@openss7.org` using the '`send-pr`' script included in the *Linux Fast-STREAMS* package, only after reading the '`BUGS`' file in the release, or See Section 9.2 [Problem Reports], page 137. You can access the OpenSS7 GNATS database directly via the web, however, the preferred method for sending new bug reports is via mail with the '`send-pr`' script.

Mailing Lists

The OpenSS7 Project provides a number of general discussion Mailing Lists for discussion concerning the *OpenSS7 Linux Fast-STREAMS* package as well as other packages released by The OpenSS7 Project.

These are `mailman` mailing lists and so have convenient web interfaces for subscribers to control their settings. See `http://www.openss7.org/mailinglist.html`.

The mailing lists are as follows:

‘openss7’ The ‘openss7’ mailing list is for general enquiries, information exchange and announcements regarding the OpenSS7 Project. This is our original mailing list and takes the highest amount of traffic.

‘openss7-announce’
The ‘openss7-announce’ mailing list is for announcements related to the OpenSS7 Project. This list will accept announcements posted by subscribers. Subscribe to this list if you are interested in announcements from the OpenSS7 Project, subscribers and sponsors, related to the OpenSS7 Project or STREAMS, SS7, SIGTRAN or SCTP in general.

‘openss7-cvs’
The ‘openss7-cvs’ mailing list is for automatic CVS log reporting. You must get permission of the owner to subscribe to this list. Subscribers are not allowed to post to this list, this is merely for distributing notification of changes to the CVS repository.h

‘openss7-develop’
The ‘openss7-develop’ mailing list is for email exchange related to the development projects under the OpenSS7 Project. This includes development requests, proposals, requests for comment or proposal. Subscribe to this list if you are interested in ongoing development details regarding the OpenSS7 Project.

‘openss7-test’
The ‘openss7-test’ mailing list is for email exchange related to the testing of code under the OpenSS7 Project. This specifically relates to conformance testing, verification testing, interoperability testing and beta testing. Subscribe to this list if you are interested in participating in and receiving ongoing details of test activities under the OpenSS7 Project.

‘openss7-bugs’
The ‘openss7-bugs’ mailing list is specifically tailored to bug tracking. The mailing list takes a feed from the OpenSS7 GNATS bug tracking system and accepts posting of responses to bug reports, tracking and resolution. Subscribe to this list if you are interested in receiving detailed *OpenSS7* release code bug tracking information. This list is not archived; for historical information on problem reports, see our GNATS databases.

‘openss7-updates’
The ‘openss7-updates’ mailing list provides updates on OpenSS7 Project code releases and ongoing activities. Subscribers are not allowed to post to this list; this list is for official OpenSS7 Project announcements only. Subscribe to this list if you are interested in receiving updates concerning official releases and activities of the OpenSS7 Project.

‘openss7-streams’
The ‘openss7-streams’ mailing list is for email exchange related to the *STREAMS* development projects under the OpenSS7 Project. This includes development requests, proposals, requests for comment or proposal. Subscribe

to this list if you are interested in ongoing development details regarding the OpenSS7 Project *STREAMS* components.

‘`linux-streams`’
: The ‘`linux-streams`’ mailing list is for mail exchange related to *Linux Fast-STREAMS* or *Linux STREAMS*. This includes patches, development requests, proposals, requests for comment or proposal. Subscribe to this list if you are interested in ongoing development details regarding the *STREAMS* for Linux components. This is the the new (September 2006) home of the ‘`linux-streams`’ list formerly of `gsyc.escet.urjc.es`.

Spam

To avoid spam being sent to the members of the *OpenSS7* mailing list(s), we have blocked mail from non-subscribers. Please subscribe to the mailing list before attempting to post to them. (Attempts to post when not subscribed get bounced.)

As an additional measure against spam, subscriber lists for all *OpenSS7* mailing lists are not accessible to non-subscribers; for most lists subscriber lists are only accessible to the list administrator. This keeps your mailing address from being picked off our website by bulk mailers.

Acceptable Use Policy

It is acceptable to post professional and courteous messages regarding the *OpenSS7* package or any general information or questions concerning *STREAMS*, *SS7*, *SIGTRAN*, *SCTP* or telecommunications applications in general.

Large Attachments

The mailing list is blocked from messages of greater than `40k`. If you have attachments (patches, test programs, etc.) and you mail them to the list, it will bounce to the list administrator. If you are interested in making your patches, test programs, test results or other large attachments available to the members of the mailing list, state in the message that you would like them posted and the list administrator will place them in the mail archives.

Quick Start Guide

Linux Fast-STREAMS

Package streams-0.9.2.4 was released under AGPLv3 2008-10-31.

The *OpenSS7 Linux Fast-STREAMS* package is a High-Performance *STREAMS* framework for *Linux* that is compatible with *SVR 4.2 MP STREAMS* and a host of other commercial *UNIX*® *STREAMS* implementations, with complete debugging and production release capabilities. It is as a high-performance, production replacement for the buggy and now deprecated *Linux STREAMS (LiS)*.

The *Linux Fast-STREAMS* package includes kernel modules, *SVR 4.2 STREAMS* drivers, modules, libraries, utilities, test programs, daemons, and development environment for the development and execution of *STREAMS* modules and drivers under *Linux Fast-STREAMS*. It is completely documented with over four hundred (435) manual pages, and three (3) major print set manuals.

The package configures, compiles, installs and builds '`rpm`'s or '`deb`'s for a wide range of Linux `rpm(1)`- or `dpkg(1)`-based distributions, and can be used on production kernels without patching, recompiling or tainting the kernel. Its small run-time footprint makes the release suitable for embedded targets.

This distribution is only currently applicable to *Linux* 2.4 and 2.6 kernels and was targeted at `ix86`, `x86_64`, `ppc` and `ppc64` architectures, but should build and install for other architectures as well.

Release

This is the `streams-0.9.2.4` package, released 2008-10-31. This '`0.9.2.4`' release, and the latest version, can be obtained from the download area of The OpenSS7 Project website using a command such as:

```
$> wget http://www.openss7.org/tarballs/streams-0.9.2.4.tar.bz2
```

The release is available as an `autoconf(1)` tarball, '`src.rpm`' or '`dsc`', as a set of binary '`rpm`'s or '`deb`'s, or as a `yum(8)` or `apt(8)` repository. See the download page for the `autoconf(1)` tarballs, '`src.rpm`'s, '`dsc`'s, or repository access instructions. See the `streams` package page for tarballs, source and binary packages.

Please see the '`NEWS`' file for release notes and history of user visible changes for the current version, and the '`ChangeLog`' file for a more detailed history of implementation changes. The '`TODO`' file lists features not yet implemented and other outstanding items.

Please see the '`INSTALL`', '`INSTALL-streams`' and '`README-make`', files (or see Chapter 8 [Installation], page 85) for installation instructions.

When working from `cvs(1)` or `git(1)`, please see the '`README-cvs`', file (or see Section 8.2.8 [Downloading from CVS], page 94). An abbreviated installation procedure that works for most applications appears below.

This release of the package is published strictly under Version 3 of the *GNU Affero Public License* which can be found in the file '`COPYING`'. Package specific licensing terms (if any) can be found in the file '`LICENSES`'. Please respect these licensing arrangements. If you

are interested in different licensing terms, please contact the copyright holder, or OpenSS7 Corporation <sales@openss7.com>.

See 'README-alpha' (if it exists) for alpha release information.

Prerequisites

The quickest and easiest way to ensure that all prerequisites are met is to download and install this package from within the *OpenSS7 Master Package*, `openss7-0.9.2.G`, instead of separately.

Prerequisites for the Linux Fast-STREAMS package are as follows:

1. *Linux* distribution, somewhat *Linux Standards Base* compliant, with a 2.4 or 2.6 kernel and the appropriate tool chain for compiling out-of-tree kernel modules. Most recent *Linux* distributions are usable out of the box, but some development packages must be installed. For more information, see Section 7.2 [Compatibility], page 61.
   - A fairly LSB compliant GNU/Linux distribution.[1]
   - Linux 2.4 kernel (2.4.10 - 2.4.27), or
   - Linux 2.6 kernel (2.6.3 - 2.6.26);
   - glibc2 or better.
   - GNU groff (for man pages).[2]
   - GNU texinfo (for info files).
   - GNU bison and flex (for config programs).
   - net-snmp (for SNMP agents).[3]

When configuring and building multiple *OpenSS7 Project* release packages, place all of the source packages (unpacked tarballs) at the same directory level and all build directories at the same directory level (e.g. all source packages under '`/usr/src`').

When installing packages that install as kernel modules, it is necessary to have the correct kernel development package installed. For the following distributions, use the following commands:

```
Ubuntu:  $> apt-get install linux-headers
Debian:  $> apt-get install kernel-headers
Fedora:  $> yum install kernel-devel
```

You also need the same version of `gcc(1)` compiler with which the kernel was built. If it is not the default, add '`CC=kgcc`' on the line after '`./configure`', for example:

```
$> ../streams-0.9.2.4/configure CC='gcc-3.4'
```

Installation

The following commands will download, configure, build, check, install, validate, uninstall and remove the package:

```
$> wget http://www.openss7.org/tarballs/streams-0.9.2.4.tar.bz2
$> tar -xjvf streams-0.9.2.4.tar.bz2
$> mkdir build
```

[1] See Section 7.2.1 [GNU/Linux Distributions], page 61, for more information.

[2] If you are using a Debian release, please make sure to install the groff extension package ('`groff_ext`'), as it contains the `refer` or `grefer` commands necessary for including references in the manual pages.

[3] A wide range of net-snmp releases are supported, from UCD-SNMP 4.2.5 through net-snmp 5.4.

```
$> pushd build
$> ../streams-0.9.2.4/configure --enable-autotest
$> make
$> make check
$> sudo make install
$> sudo make installcheck
$> sudo make uninstall
$> popd
$> sudo rm -rf build
$> rm -rf streams-0.9.2.4
$> rm -f streams-0.9.2.4.tar.bz2
```

If you have problems, try building with the logging targets instead. If the make of a logging target fails, an automatic problem report will be generated that can be mailed to The OpenSS7 Project.[4] Installation steps using the logging targets proceed as follows:

```
$> wget http://www.openss7.org/tarballs/streams-0.9.2.4.tar.bz2
$> tar -xjvf streams-0.9.2.4.tar.bz2
$> mkdir build
$> pushd build
$> ../streams-0.9.2.4/configure --enable-autotest
$> make compile.log
$> make check.log
$> sudo make install.log
$> sudo make installcheck.log
$> sudo make uninstall.log
$> popd
$> sudo rm -rf build
$> rm -rf streams-0.9.2.4
$> rm -f streams-0.9.2.4.tar.bz2
```

See '`README-make`' for additional specialized make targets.

For custom applications, see the '`INSTALL`' and '`INSTALL-streams`' files or the see Chapter 8 [Installation], page 85, as listed below. If you encounter troubles, see Chapter 9 [Troubleshooting], page 133, before issuing a bug report.

Brief Installation Instructions

The Linux Fast-STREAMS package is available from the downloads area of The OpenSS7 Project website using a command such as:

```
$> wget http://www.openss7.org/tarballs/streams-0.9.2.4.tar.bz2
```

Unpack the tarball using a command such as:

```
$> tar -xjvf streams-0.9.2.4.tar.bz2
```

The tarball will unpack into the relative subdirectory named after the package name: streams-0.9.2.4.

The package builds using the GNU `autoconf` utilities and the '`configure`' script. To build the package, we recommend using a separate '`build`' directory as follows:

[4] Please see Section 9.2 [Problem Reports], page 137, or the file '`PROBLEMS`' in the release directory for more information on filing a proper *Problem Report*.

```
$> mkdir build
$> cd build
$> ../streams-0.9.2.4/configure
```

In general, the package configures and builds without adding any special options to the '`configure`' script. For general options to the '`configure`' script, see the GNU '`INSTALL`' file in the distribution:

```
$> less ../streams-0.9.2.4/INSTALL
```

For specific options to the '`configure`' script, see the '`INSTALL-streams`' file in the distribution, or simply execute the configure script with the '`--help`' option like so:

```
$> ../streams-0.9.2.4/configure --help
```

After configuring the package, the package can be compiled simply by issuing the '`make`' command:

```
$> make
```

Some specialized makefile targets exists, see the '`README-make`' file in the distribution or simply invoke the '`help`' target like so:

```
$> make help | less
```

After successfully building the package, the package can be checked by invoking the '`check`' make target like so:

```
$> make check
```

After successfully checking the package, the package can be installed by invoking the '`install`' make target (as root) like so:

```
$> sudo make install
```

The test suites that ship with the package can be invoked after the package has been installed by invoking the '`installcheck`' target. This target can either be invoked as root, or as a normal user, like so:

```
$> make installcheck
```

(Note: you must add the '`--enable-autotest`' flag to '`configure`', above for the test suites to be invoked with '`make installcheck`'.)

The package can be cleanly removed by invoking the '`uninstall`' target (as root):

```
$> sudo make uninstall
```

Then the build directory and tarball can be simply removed:

```
$> cd ..
$> rm -rf build
$> rm -rf streams-0.9.2.4
$> rm -f streams-0.9.2.4.tar.bz2
```

Detailed Installation Instructions

More detailed installation instructions can be found in the Chapter 8 [Installation], page 85, contained in the distribution in '`text`', '`info`', '`html`' and '`pdf`' formats:

```
$> cd ../streams-0.9.2.4
$> less doc/manual/streams.txt
$> lynx doc/manual/streams.html
```

```
$> info doc/manual/streams.info
$> xpdf doc/manual/streams.pdf
```

The ‘`text`’ version of the manual is always available in the ‘`MANUAL`’ file in the release.

The current manual is also always available online from *The OpenSS7 Project* website at:

```
$> lynx http://www.openss7.org/streams_manual.html
```

1 Introduction

This manual documents the design, implementation, installation, operation and future development schedule of the *Linux Fast-STREAMS* package.

1.1 Overview

This manual documents the design, implementation, installation, operation and future development of the *Linux Fast-STREAMS* package.

1.2 Organization of this Manual

This manual is organized (loosely) into several sections as follows:

1.3 Conventions and Definitions

This manual uses *texinfo* typographic conventions.

2 Objective

2.1 Background

STREAMS derives from Dennis Ritchie's original paper,[1] was incorporated into the *UNIX® System V Release 3* operating system, replaced the terminal input-output subsystem, pipes and FIFOs in *UNIX® System V Release 4*, and was improved in the USL release of the *UNIX® System V Release 4.2* operating system.

Today, STREAMS is a part of every major branded *UNIX®* variant, such as *AIX®*, *HP-UX®*, *IRIX®*, *MacOT®*, *OSF/1®*, *Solaris®*, *SUPER-UX®*, *UnixWare®*, *UXP/V®*, and including many *UNIX*-like operating systems and popular embedded RTOS, but with the notable exception of *Berkeley System Distribution* releases, variants and offshoots, and *Linux*.

2.2 What is STREAMS?

STREAMS is a flexible framework for communication between a user level process and a kernel resident driver. It encompasses a set of kernel system calls providing a user-kernel interface that is backward compatible with the traditional character device driver interface, as well as a set of STREAMS driver and module entry points forming a driver-kernel interface. STREAMS also provides a rich set of kernel utility functions for the development and implementation of kernel-resident drivers and modules. STREAMS prompted the specification of the DDI/DKI which is an architecture independent driver-kernel interface that provides a standardized set of kernel functions (beyond just STREAMS) for the development of device and software drivers.

STREAMS provides a reconfigurable full-duplex communications path between user level process and kernel resident driver, termed a *Stream*. Modules can be inserted in the path between the user and driver under user level control. Streams can be linked across multiplexers under user control to form complex (yet reconfigurable) topologies of user level processes and drivers.

Communication of control and data information along a Stream is accomplished by message passing. There is no direct function call interface between components of a Stream. A Stream exists within the STREAMS framework inside the kernel and extend from the user-kernel interface to the kernel driver interface. Each component of a Stream consists of a pair of queues used to pass messages in the upstream direction to the kernel-user interface; or downstream, the kernel-driver interface.

At the kernel-user end of the Stream is a component called the *Stream head*. As with all components of a Stream, the Stream head consists of a queue pair and a specialized set of procedures. The Stream head procedures are responsible for converting between the system call interface presented to users and the message passing mechanism within the Stream.

At the kernel-driver end of the Stream is the *Stream end*. The Stream end also contains a queue pair and a set of procedures. The Stream end (or simply driver) procedures are

[1] "A Stream Input-Output System", *AT&T Bell Laboratories Technical Journal* **63**, No. 8 Part 2 (October, 1984), pp. 1897-1910.

responsible for converting between the message passing mechanism within the Stream and the actions and events of a hardware (or pseudo-) device.

Intermediate components within the Stream are called *Modules*. Modules consist of a queue pair for passing messages upstream and downstream, as well as a set of procedures for processing messages. Modules can be pushed onto the module stack between the Stream head and Stream end using a set of standardized input-output control commands.

In support of topologies more complex than these simple linear segments, STREAMS also provides a specialized Stream end (driver) called a *Multiplexing driver*. A Multiplexing driver has the ability to open multiple Streams to its upper interface (multiplexer) as well as linking multiple Streams beneath its lower interface (multiplexer). Again, a standardized set of input-output controls provide the user with the ability to configure a Multiplexing driver.

2.3 Why STREAMS?

With the ability to open multiple Streams to a driver, push and pop modules to and from the module stack on a Stream, and to link any Stream under a multiplexing driver–all under user control using standardized input-output controls–allows STREAMS to configure complex topologies to form protocol stacks.

Almost all specialized standard telecommunications software developed since 1990 was developed to run on STREAMS. This is for several reasons:

- Since 1990, STREAMS and the associated DDI/DKI has been, and remains, the only way to incorporate OEM protocol stacks into mainstream *UNIX*® system kernels.
- The original *UNIX System Laboratories* (later *X/Open* then later the *OpenGroup*) support for ITU-T developed OSI protocols, makes STREAMS amenable to an open model for development for ITU-T protocols. (ITU-T, formerly CCITT, is the *International Telecommunications Union – Telephone Sector* responsible for international telephone standards, and original developers of the OSI model.)

As a result, there is a significant body of commercial software implementing telecommunications protocol stacks that was developed, tested, validated, conformance tested, field verified, to run on STREAMS: and is still running on STREAMS.

The cost of reimplementation, retesting, revalidation, redoing conformance testing, and field re-verification, would likely be prohibitive: after all, the point of *Linux* is reducing cost, is it not?

2.4 Why STREAMS for Linux?

The *Linux* kernel was not developed with STREAMS in mind. For TPI/IP networking, *Linux* originally followed in the footsteps of the BSD NET2 release. Currently, the implementation of TCP/IP in the *Linux* kernel has long departed from the classical BSD organization and exhibits characteristics unique to the *GNU/Linux* operating system. For character device and terminal input-output, *Linux* follows closely the SVR 3 pre-STREAMS approach to pipes, FIFOs and terminal subsystem. The terminal subsystem implementation, too, has become unique to *GNU/Linux*.

Therefore, from the perspective of TCP/IP networking and Terminal I/O, there would be little reason to provide STREAMS for *Linux*. That is, if it were not for the body of software

supporting OSI and telecommunications protocols based solely on STREAMS, for which *Linux* has little or no support.

So, the answer to the question, "Why STREAMS for Linux?" is: so that a *GNU/Linux* platform can enjoy the same wealth of telecommunications and OSI protocol stacks otherwise only available to big-iron *UNIX*®. Without STREAMS, *Linux* is probably just another BSD, and probably not a very good one.

2.5 History of STREAMS for Linux

In the mid-90's, GCOM embarked on development on an open source implementation of STREAMS called *Linux STREAMS (LiS)*, likely driven by its use for porting existing OSI protocol stacks to *Linux*. In 2000, The OpenSS7 Project abandoned using the *Linux* networking model for implementation of the *Signalling System No. 7* protocol (primarily due to the lack of support for the full BSD networking model under *Linux*) and switched to using STREAMS as the basis for all future development. The *GCOM LiS* release (2.2 at the time) was used as the STREAMS package. Over the span of the next 5 years, (and not surprisingly given the body of software), almost all *Signalling System No. 7* products released on *Linux* used *LiS* for STREAMS. In 2005, Dave Grothe (the *G* in *GCOM*) announced that he would no longer be maintaining or developing *LiS* subsequent to the 2.18.0 release, stranding many users of the package.

Later in 2005, after briefly maintaining two GPL'ed releases of *LiS*, (2.18.1 and 2.18.2), The OpenSS7 Project release (after two years of development) the '`streams-0.7a.4`' package: a reimplementation of *SVR 4.2 STREAMS* with compatibility modules for all major *UNIX*® releases, called *Linux Fast-STREAMS*. *Linux Fast-STREAMS* was intended as a *POSIX/SUSv3 XSR* conforming, high performance, production grade, replacement for *LiS*, suitable for mainline *Linux* adoption, and a better foundation on which to base SIGTRAN, VoIP, ISDN and SS7 protocol stacks developed under the The OpenSS7 Project, as well as a better foundation for porting commercial *UNIX*® OEM implementations to *Linux*. It is the '`streams-0.9.2.4`' package that contains the documentation you are reading now.

2.6 Why Fast?

After working with *LiS* releases for over 3 years, in late 2003, The OpenSS7 Project decided to begin implementation of a replacement for *LiS*, because of a number of shortcomings of the *LiS* releases:

a. unsuitable for mainline kernel adoption due to coding style and organization;
b. poorly adapted to distribution production kernels;
c. is unsuitable for packaging and repeatability;
d. portability objective unsuitable for mainline kernel adoption;
e. ports form the same baseline obfuscate the code;
f. performs poorly due to portability and coding style;
g. code bloat and over sized memory footprint;
h. redundant debug statements obscuring defects and obfuscating code;
i. overuse of semaphores;
j. contains serious races and not suitable for threaded applications;

k. does not conform to mainstream *UNIX*® implementations;
l. does not conform to *POSIX* or any release of the *Single UNIX Specification*;
m. limited set of standard drivers and modules;
n. limited set of diagnostic and administrative utilities;
o. limited test programs;
p. poorly documented.

The replacement, named *Linux Fast-STREAMS*, was to correct all of these difficulties, and, by the initial '`streams-0.7a.4`' release, was:

a. completely Lindented and follows kernel coding practises;
b. automatically adapts to production kernels with autoconf;
c. packages itself into LSB compliant RPMs and DEBs;
d. designed and implemented specifically for *GNU/Linux*;
e. no ports considered;
f. over twice the performance;
g. less than one-eighth of the memory footprint;
h. proper programming by assertion;
i. proper use of lightweight spin locks;
j. race free locking strategies and synchronization;
k. compatible with all mainstream *UNIX*® implementations;
l. conforms to *POSIX/SUSv3 XSR*;
m. complete set of standard drivers and modules;
n. complete set of diagnostic and administrative utilities;
o. integrated set of conformance test suites;
p. fully documented.

Many specific difficulties encountered with *LiS* not repeated by *Linux Fast-STREAMS* are contained in the *COMPATIBILITY* section of most of the manual pages.

3 Reference

3.1 Files

The following kernel modules are installed by *Linux Fast-STREAMS* in the '`/lib/modules/2.4.20-28.7/streams/`'directory, with either a '`.o`' or '`.ko`' extension.[1]

'`specfs`'
: This kernel module contains the *STREAMS* Special Shadow Filesystem. See `specfs(5)` for more information.

'`streams`'
: This kernel module contains the *STREAMS* scheduler, utility functions, and *STREAMS* Device Driver Interface/Driver Kernel Interface (DDI/DKI). See `STREAMS(9)` for more information.

'`streams-fifo`'
: This kernel module contains the '`fifo`' *STREAMS* driver. This is a standard *STREAMS* driver, but is also used by the conformance and validation test suite. See `fifo(4)` for more information.

'`streams-sad`'
: This kernel module contains the '`sad`' *STREAMS* driver. This is the standard *STREAMS* Administrative Driver. See `sad(4)` for more information.

'`streams-nsdev`'
: This kernel module contains the '`nsdev`' *STREAMS* driver. This is a *Linux Fast-STREAMS* specific driver. See `nsdev(4)` for more information.

'`streams-echo`'
: This kernel module contains the '`echo`' *STREAMS* driver. This is a standard *STREAMS* driver, but is also used by the conformance and validation test suite. See `echo(4)` for more information.

'`streams-mux`'
: This kernel module contains the '`mux`' *STREAMS* driver. This is a standard *STREAMS* driver< but is also used by the conformance and validation test suite. See `mux(4)` for more information.

'`streams-nuls`'
: This kernel module contains the '`nuls`' *STREAMS* driver. This is a standard *STREAMS* module. See `nuls(4)` for more information.

'`streams-pipe`'
: This kernel module contains the '`pipe`' *STREAMS* driver. This is a standard *STREAMS* driver. See `pipe(4)` for more information.

'`streams-log`'
: This kernel module contains the '`log`' *STREAMS* driver. This is a standard *STREAMS* driver. See `log(4)` for more information.

[1] The kernel version, 2.4.20-28.7 is just an example. When installed with a '`.ko`' extension, dashes ('`-`') are also changed to underscores ('`_`').

‘`streams-loop`’
: This kernel module contains the ‘`loop`’ *STREAMS* driver. This is a standard *STREAMS* driver, but is also used by the conformance and validation test suite. See `loop(4)` for more information.

‘`streams-sfx`’
: This kernel module contains the *sfx STREAMS* driver. This is a common character device driver for implementing *STREAMS* FIFOs. See `sfx(4)` for more information.

‘`streams-spx`’
: This kernel module contains the *spx STREAMS* driver. This is a common character device driver for implementing *STREAMS* pipes. See `spx(4)` for more information.

‘`streams-bufmod`’
: This kernel module contains the ‘`bufmod`’ *STREAMS* module. The ‘`bufmod`’ *STREAMS* module is a simple buffer module (a module that always defers to its service procedure and then passes any message along). This module is used for performance testing of the *STREAMS* package. See `bufmod(4)` for more information.

‘`streams-nullmod`’
: This kernel module contains the ‘`nullmod`’ *STREAMS* module. The ‘`nullmod`’ *STREAMS* module is a simple null module (a module that always passes messages to the next module in along the Stream). This module is used for performance testing of the *STREAMS* package and is also used by the conformance and validation test suite. See `nullmod(4)` for more information.

‘`streams-pipemod`’
: This kernel module contains the ‘`pipemod`’ *STREAMS* module. This is a standard *STREAMS* module used with pipes. See `pipemod(4)` for more information.

‘`streams-connld`’
: This kernel module contains the ‘`connld`’ *STREAMS* module. This is a standard *STREAMS* module. See `connld(4)` for more information.

‘`streams-sc`’
: This kernel module contains the *sc STREAMS* module. This is a common *STREAMS* Configuration module. See `sc(4)` for more information.

‘`streams-testmod`’
: This kernel module contains the ‘`testmod`’ *STREAMS* module. This is a *Linux Fast-STREAMS* specific test module that is used for conformance and validation testing of *STREAMS*. See `testmod(4)` for more information.

Additional kernel modules are provided by add-on packages.

3.2 Drivers

The configuration of *STREAMS* drivers and modules is performed when compiling the *Linux Fast-STREAMS* subsystem. The *STREAMS* subsystem, core drivers and modules are part of every *Linux Fast-STREAMS* system.

The following lists the core drivers and modules, *STREAMS* kernel tunable parameters, and *STREAMS* configuration information:[2]

`clone(4)` (‘`streams`’)
: Clone device driver. This is a standard *SVR 4.2 STREAMS* driver. The `clone(4)` driver is a integral part of *STREAMS* and is used to create clone instances of a *STREAMS* driver.

  See `clone(4)` for more information.

`echo(4)` (‘`streams-echo`’)
: Echo (loopback) device driver. This is a commonly implemented *STREAMS* driver. It is implemented by *HP-UX*® and *OSF/1*®. The `echo(4)` driver provides a simple FIFO-like device without full *POSIX* FIFO semantics. Its primary purpose is for the *STREAMS* Verification function, `strvf(8)`, and the `test-streams(8)` validation test suite.

  See `echo(4)` for more information.

`fifo(4)` (‘`streams-fifo`’)
: FIFO (Named Pipe) device driver. This is a standard *SVR 4.2 STREAMS* driver. The `fifo(4)` driver provides *POSIX*-compliant *STREAMS*-based *FIFO* device. Not all implementations of *STREAMS* provide *STREAMS*-based *FIFO*s: some implementations use the older *SVR 3*-style *FIFO*s that are not *STREAMS*-based. *Linux Fast-STREAMS* provides *STREAMS*-based *FIFO*s with the `fifo(4)` driver.

  See `fifo(4)` for more information.

`log(4)` (‘`streams-log`’)
: *STREAMS* log driver. This is a standard *SVR 4.2 STREAMS* driver. The `log(4)` driver provides a *STREAMS* capable logger in addition to the *BSD* logger present in *Linux*. The `log(4)` driver provides additional support for *STREAMS* modules and drivers using the `strlog(9)` kernel level utility. *Linux Fast-STREAMS* also provides the `strace(8)`, `strerr(8)` and `strclean(8)` administrative utility functions and startup scripts for controlling the `log(4)` driver.

  See `log(4)` for more information.

`loop(4)` (‘`streams-loop`’)
: Loop device driver. This is a standard *SVR 4.2 STREAMS* driver. The `loop` driver is detailed in the *UNIX System V Release 4 Programmer's Manual – STREAMS*. The `loop(4)` driver provides capabilities used primarily for validation test programs (see `test-streams(8)`) as well as serving as an example driver.

[2] Note that documentation for *STREAMS* drivers is in the form of manual pages in Section 4 of the manual pages with the same name as the driver.

See `loop(4)` for more information.

`mux(4)` (‘`streams-mux`’)
: Multiplexing driver. This is a standard *SVR 4.2 STREAMS* driver. The `mux` driver is detailed in the *UNIX System V Release 4 Programmer's Manual – STREAMS*. The `mux(4)` driver provides capabilities used primarily for validation test programs (see `test-streams(8)` as well as serving as an example multiplexing driver. This `mux(4)` driver also provides the `minimux` capabilities formerly present in *LiS*.

  See `mux(4)` for more information.

`nsdev(4)` (‘`streams-nsdev`’)
: Named *STREAMS* device driver. This is a *Linux Fast-STREAMS* specific driver. The `nsdev(4)` driver is a `clone(4)`-like driver that permits the specification of major and minor device numbers using the device node name. It provides one of three mechanisms under *Linux Fast-STREAMS* that remove *STREAMS* driver dependency on statically allocated device numbers.

  See `nsdev(4)` for more information.

`nuls(4)` (‘`streams-nuls`’)
: Null *Stream* driver. This is a standard *SVR 4.2 STREAMS* driver. The `nuls(4)` driver is usually called ‘`null`’. *Linux* has its own *SVR3*-style ‘`/dev/null`’ driver, so it was renamed to ‘`nuls`’.

  See `nuls(4)` for more information.

`pipe(4)` (‘`streams-pipe`’)
: *STREAMS*-based pipe driver. This is a standard *SVR 4.2 STREAMS* driver. However, `pipe(4)` is not normally implemented as a *STREAMS* driver, but is implemented as a system call. *Linux Fast-STREAMS* provides `pipe(2s)` system call emulation which invokes this driver internal to the kernel.

  See `pipe(4)` for more information.

`sad(4)` (‘`streams-sad`’)
: *STREAMS* Administrative Driver. This is a standard *SVR 4.2 STREAMS* driver. The `sad(4)` driver is used by the `autopush(8)` utility to examine and specify the autopush lists for *STREAMS* drivers. Also, it is used to examine and verify the present of *STREAMS* modules or drivers in the system.

  See `sad(4)` for more information.

`sfx(4)` (‘`streams-sfx`’)
: *STREAMS* FIFO device driver. This is commonly implemented *STREAMS* driver that is used to implement *STREAMS* FIFOs (Named Pipes) using a regular character device. The `sfx(4)` driver provides a character based device approach to creating FIFOs.

  See `sfx(4)` for more information.

`spx(4)` (‘`streams-spx`’)
: *STREAMS* pipe device driver. This is commonly implemented *STREAMS* driver that is used to implement *STREAMS* pipes using a regular character de-

vice. The spx(4) driver provides a character based device approach to creating FIFOs and pipes. Only *UnixWare*® and *AIX(4)* document this device.

See spx(4) for more information.

Additional drivers are provided by add-on packages.

3.3 Modules

The configuration of *STREAMS* drivers and modules is performed when compiling the *Linux Fast-STREAMS* subsystem. The *STREAMS* subsystem, core drivers and modules are part of every *Linux Fast-STREAMS* system.

The following lists the core drivers and modules, *STREAMS* kernel tunable parameters, and *STREAMS* configuration information:[1]

`pipemod(4)` (‘`streams-pipemod`’)
: Pipe module. This is a standard *SVR 4.2 STREAMS* module. The `pipemod(4)` module can be pushed over a pipe end or FIFO before other modules are pushed (on either end) to reverse the sense of the `M_FLUSH(9)` message that traverse the pipe.

  See `pipemod(4)` for more information.

`connld(4)` (‘`streams-connld`’)
: Connection Line Discipline module. This is a standard *SVR 4.2 STREAMS* module. The `connld(4)` module can be pushed over a pipe end that has been attached to a file system file using `fattach(3)` and will then create a new pipe instance on each `open(2s)` of the attached file and pass the new remove file pointer to the remove end using `M_PASSFP(9)` to be received with `I_RECVFD(7)`. This allows servers to be created that use `pipe(4)`s for communication.

  See `connld(4)` for more information.

`sc(4)` (‘`streams-sc`’)
: *STREAMS* Configuration module. This is a commonly implemented *STREAMS* module. It is implemented by *HP-UX* and *AIX*, and perhaps other *Mentat*-derived *STREAMS* implementations. The `sc(4)` modules provides the ability to access *STREAMS* driver information by name rather than major device number. It also provides access to the `module_info(9)` and `module_stat(9)` structure information for the named *STREAMS* module or driver, not accessible using the `sad(4)` driver. The `sc(4)` module is used by the `scls(8)` utility.

  See `sc(4)` for more information.

`bufmod(4)` (‘`streams-bufmod`’)
: Buffer module. This is a standard *SVR 4.2 STREAMS* module described in the *UNIX System V Release 4 Programmer's Manual – STREAMS*. The `bufmod(4)` module also has *Linux Fast-STREAMS* specific extensions. The `bufmod(4)` module is used by the `perftest(8)` performance test program to test the effect of additional levels of service procedure pushed over a *Stream*. The module also serves as an example of a *STREAMS* module using service procedures.

  See `bufmod(4)` for more information.

`nullmod(4)` (‘`streams-nullmod`’)
: Null module. This is a standard *SVR 4.2 STREAMS* module described in the *UNIX System V Release 4 Programmer's Manual – STREAMS*. The

[1] Note that documentation for *STREAMS* modules is in the form of a manual page in Section 4 of the manual pages with the same name as the module.

`nullmod(4)` module also has *Linux Fast-STREAMS* specific extensions. The `nullmod(4)` module is used by the `perftest(8)` performance test program to test the effect of additional levels of put procedure pushed over a *Stream*. The module also serves as an example of a *STREAMS* module not using service procedures.

See `nullmod(4)` for more information.

`testmod(4)` (‘`streams-testmod`’)
Test module. This is a *Linux Fast-STREAMS* specific *STREAMS* module. The primary purpose of the `testmod(4)` modules is to provide the `test-streams(8)` validation test program with the capability to pass specific `M_ERROR(9)` and `M_HANGUP(9)` messages to the *Stream head* for *POSIX* validation testing. It also serves as an example of how a *STREAMS* module can properly process `M_IOCTL(9)` and related messages.

See `testmod(4)` for more information.

Additional modules are provided by add-on packages.

3.4 Libraries

During the installation process of *Linux Fast-STREAMS* a subroutine library is built and installed on your system. For 64-bit systems that support 32-bit compatibility, two versions of each library are built and installed: one 64-bit native library and one 32-bit compatibility library. 64-bit native libraries are installed to the '`/usr/lib64`' subdirectory. 32-bit native and 32-bit compatibility libraries are installed to the '`/usr/lib`' subdirectory.

'`libstreams.so.0.0.1`'
'`libstreams.so.0`'
'`libstreams.so`'
: Provides a shared object library for use by *STREAMS* applications programs.

'`libstreams.a`'
: Provides a static library for use by *STREAMS* applications programs.

'`libstreams.la`'
: Provides the `libtool` definitions for the library.

In addition to the '`libstreams`' library, *Linux Fast-STREAMS* also installs compatibility libraries for *LiS*. These compatibility libraries permit applications previously linked with *LiS* shared libraries to function with *Linux Fast-STREAMS* without recompiling or relinking.

'`libLiS.so.0.0.1`'
'`libLiS.so.0`'
'`libLiS.so`'
: Provides a shared object library for use by legacy *LiS* applications programs.

'`libLiS.a`'
: Provides a static library for use by legacy *LiS* applications programs.

'`libLiS.la`'
: Provides the `libtool` definitions for the library.

'`libpLiS.so.0.0.1`'
'`libpLiS.so.0`'
'`libpLiS.so`'
: Provides a shared object library for use by legacy *LiS* applications programs.

'`libpLiS.a`'
: Provides a static library for use by legacy *LiS* applications programs.

'`libpLiS.la`'
: Provides the `libtool` definitions for the library.

3.4.1 libstreams Library Routines

The following routines are present in the '`libstreams`' libraries. The routines in these libraries are standard *STREAMS* interface system calls documented in the *System V Release 4.2 Programmer's Manual – STREAMS*. Refer to the associated manual pages for detailed information on these routines.

`fattach(2)`
: Name a *STREAMS* special file.

`fdetach(2)`
: Unname a *STREAMS* special file.

`getmsg(2)`
: Get next message off of a Stream.

`getpmsg(2s)`
: Get next message off of a Stream.

`isastream(2)`
: Test for a *STREAMS* special file.

`pipe(2s)` Create a *STREAMS* pipe.

`putmsg(2)`
: Put a message to a *STREAMS* character device.

`putpmsg(2s)`
: Put a band message to a *STREAMS* character device.

`pstrlog(3)`
: Print a *STREAMS* log buffer.

`strlog(3)`
: Print a *STREAMS* log buffer.

`vstrlog(3)`
: Print a *STREAMS* log buffer.

3.4.2 libLiS Library Routines

The following routines are present in the '`libLiS`' libraries. The routines are identical to the routines present in the '`libstreams`' library and are provided in the '`libLiS`' library for compatibility with existing applications linked against '`libLiS`'.

`fattach(2)`
: Name a *STREAMS* special file.

`fdetach(2)`
: Unname a *STREAMS* special file.

`getmsg(2)`
: Get next message off of a Stream.

`getpmsg(2s)`
: Get next message off of a Stream.

`isastream(2)`
: Test for a *STREAMS* special file.

`pipe(2s)` Create a *STREAMS* pipe.

`putmsg(2)`
: Put a message to a *STREAMS* character device.

`putpmsg(2s)`
: Put a band message to a *STREAMS* character device.

3.4.3 libpLiS Library Routines

The following routines are present in the '`libpLiS`' libraries. The '`libpLiS`' library is the same as the '`libLiS`' and '`libstreams`' libraries but omits the `pipe(2s)` subroutine. The purpose of the '`libpLiS`' library was to permit it to be used as a library preload without affecting the `pipe(2s)` function used by existing programs linked against '`libc`'.

`fattach(2)`
: Name a *STREAMS* special file.

`fdetach(2)`
: Unname a *STREAMS* special file.

`getmsg(2)`
: Get next message off of a Stream.

`getpmsg(2s)`
: Get next message off of a Stream.

`isastream(2)`
: Test for a *STREAMS* special file.

`putmsg(2)`
: Put a message to a *STREAMS* character device.

`putpmsg(2s)`
: Put a band message to a *STREAMS* character device.

3.4.4 Using the Library

To use one of the *Linux Fast-STREAMS* libraries you can include the file '`sys/stropts.h`' in you application program source code. On you compiler command line, add the option '`-I/usr/include/streams`' to include the version of '`sys/stropts.h`' that is distributed with *Linux Fast-STREAMS*.

When linking our program, or performing a final `gcc` to build your executable, include one of the following options on your command line:

'`/usr/lib/libstreams.a`'
'`-lstreams -static`'
: Link against the static version of the library.

'`-lstreams`'
: Link against the shared object version of the library.

'`/usr/lib/libstreams.la`'
: Use with `libtool` to link additional convenience libraries against the shared or static versions of the library.

Failure to link the executable runtime path for '`libstreams`' will result in linker-loader warnings that the functions `getpmsg(2s)` or `putpmsg(2s)` are not implemented and will always fail.[1].

See also Chapter 4 [Development], page 37 for more information.

[1] These warnings are generated when linking only against the '`libc`' library that includes only failing stubs for `getpmsg(2)` and `putpmsg(2)`

3.5 Utilities

3.5.1 Init Scripts

Following are System V Init Scripts that are installed by the package:

`specfs(8)` (‘`/etc/init.d/specfs`’)
`specfs.sh(8)` (‘`/etc/init.d/specfs.sh`’)
: System V Init Script for the *STREAMS* Special Shadow Filesystem. The `specfs(8)` init script provides the ability to initialize, configure and mount the *STREAMS* Special Shadow Filesystem, `specfs(5)`. The `specfs(8)` script provides the RedHat-style init script, whereas the `specfs.sh(8)` script provides the Debian-style init script.

  See `specfs(8)` for more information.

`streams(8)` (‘`/etc/init.d/streams`’)
`streams.sh(8)` (‘`/etc/init.d/streams.sh`’)
: System V Init Script for the *STREAMS* Subsystem. The `streams(8)` init script provides the ability to initialize, configure and mount the *STREAMS* subsystem, `STREAMS(9)`. The `streams(8)` script provides the RedHat-style init script, whereas the `streams.sh(8)` script provides the Debian-style init script.

  See `streams(8)` for more information.

3.5.2 User Utilities

Following are user utilities for manipulating *Streams*:

`strchg(1)` (‘`/usr/bin/strchg`’)
: Change *Stream* configuration. `strchg(1)` is a standard *SVR 4.2 STREAMS* user utility.

  `strchg(1)` is a C-language user program that can be used to alter the configuration of the *Stream* associated with the caller’s standard input. The `strchg(1)` command pushes modules on the *Stream*, pops modules off of the *Stream*, or both. Only the superuser or owner of the *STREAMS* device can alter the configuration of that *Stream*. If another user attempts to alter the configuration, the `strconf(1)` command will fail.

  `strchg(1)` is useful from the shell and, when standard input is redirected from an open file descriptor to the command, can be used to push and pop modules from arbitrary *Streams*, not just those associated with *STREAMS*-based terminal devices.

  See `strchg(1)` for more information.

`strconf(1)` (‘`/usr/bin/strconf`’)
: Query *Stream* configuration. `strconf(1)` is a standard *SVR 4.2 STREAMS* user utility.

  `strconf(1)` is a C-language user program that can be used to query the configuration of a *Stream*. When use without any options, it prints a list of the modules in the *Stream* associated with the standard input, as well as the topmost driver. The list is printed with one name per line, where the first name

printed is the topmost module on the *Stream* and the last item printed is the name of the topmost driver associated with the *Stream*.

`strconf(1)` is useful from the shell and, when standard input is redirected from an open file descriptor to the command, can be used to query arbitrary *Streams*, not just the associated with *STREAMS*-based terminal devices.

See `strconf(1)` for more information.

`strreset(1)` (‘`/usr/bin/strreset`’)
Reset a *Stream*. `strreset(1)` is a standard *SVR 4.2 STREAMS* user utility.

`strreset(1)`is a C-language user program that resets an open *Stream* by generating an `M_FLUSH(9)` message to the *Stream head*. It is used mainly to reset blocked *Streams*. Wehn it is impossible to reopen the *Stream*, issue an `I_FLUSH` or equivalent command. This situation may happen with a process sleeping in a module's close routine, when signals can not be sent to the process (a zombie process exiting, for example).

See `strreset(1)` for more information.

3.5.3 Administrative Utilities

Following are administrative utilities for manipulating and examining the *STREAMS* subsystem:

`autopush(8)` (‘`/usr/sbin/autopush`’)
Control the *autopush* module list for a *STREAMS* device. `autopush(8)` is a standard *SVR 4.2 STREAMS* administrative utility.

`autopush(8)` is a C-language program that can be used to manipulate and examine which *STREAMS* modules are automatically pushed over a device when it is opened. It is also possible to restrict the ability to push further modules on the *Stream* without proper privilege. The `autopush(8)` utility provides a user program interface to the *STREAMS* Administrative Driver (`sad(4)`).

See `autopush(8)` for more information.

`fattach(8)` (‘`/usr/sbin/fattach`’)
Name a *STREAMS* file. `fattach(8)` is an *LiS* utility. Although *OSF/1* documentation mentions an `fattach` manual page in section 8, one does not exist.

`fattach(8)` opens a `pipe(4)` and attaches one end of the pipe to a file using `fattach(3)`, and optionally pushes the `connld(4)` module on the side of the pipe being attached to the file. The other end of the pipe remains available for use by the shell program invoking this command.

`fattach(8)` provides a easy way for shell programs to use *STREAMS*-based pipes and to use the facilities of the `connld(4)` module.

See `fattach(8)` for more information.

`fdetach(8)` (‘`/usr/sbin/fdetach`’)
Unlink a named *STREAMS* file.

`fdetach(8)` is a standard *SVR 4.2 STREAMS* administrative utility.

`fdetach(8)` is a C-language program that detaches or disassociates a file descriptor for an open *STREAMS* device or pipe from its filename in the file system.

See `fdetach(8)` for more information.

`insf(8)` (‘`/usr/sbin/insf`’)
Install special files. `insf(8)` is the *HP-UX* way to install special (device) files. This program is not even partially implemented in *Linux Fast-STREAMS*. Use `streams_mknod(8)` and friends instead.

See `insf(8)` for more information.

`scls(8)` (‘`/usr/sbin/scls`’)
List *STREAMS* configuration. `scls(8)` is a rather useful *AIX* administrative utility that is also implemented by *Linux Fast-STREAMS*.

`scls(8)` is a C-language program that can be used to list module and driver names as well as information and statistics associated with those modules or drivers. The `scls(8)` utility provides a user program interface to the *STREAMS* Configuration module (`sc(4)`).

See `scls(8)` for more information.

`strace(8)` (‘`/usr/sbin/strace`’)
Write *STREAMS* event trace messages to the standard output. `strace(8)` is a standard *SVR 4.2 STREAMS* administrative utility.

The `strace(8)` C-language program receives trace event messages from the *STREAMS* log driver (`log(4)`) and writes these messages to the standard output. When run as a daemon, `strace(8)` appends these messages to a log file.

Messages that appear in the trace log are intended to report debugging information that assists with troubleshooting a running *STREAMS* module or driver.

See `strace(8)` for more information.

`strclean(8)` (‘`/usr/sbin/strclean`’)
Clean up after the *STREAMS* error logger. `strclean(8)` is a standard *SVR 4.2 STREAMS* administrative utility.

The `strclean(8)` utility is a bash script that can be used to delete aged log files generated by the *STREAMS* error logger, `strerr(8)`.

See `strclean(8)` for more information.

`streams_mknod(8)` (‘`/usr/sbin/streams_mknod`’)
Make special device nodes for *STREAMS*. `streams_mknod(8)` is a *Linux Fast-STREAMS* specific administrative utility.

The `streams_mknod(8)` C-language program can be used to make (or remove) the special device nodes under the ‘`/dev`’ directory required by `streams-0.9.2.4` package modules and drivers. `streams_mknod(8)` in invoked by the System V startup script, ‘`/etc/init.d/streams`’.

See `streams_mknod(8)` for more information.

`strerr(8)` (‘`/usr/sbin/strerr`’)
: Receive error log messages from the *STREAMS* `log(4)` driver. `strerr(8)` is a standard *SVR 4.2 STREAMS* administrative utility.

  The `strerr(8)` utility is a C-language program, run as a daemon, that receives error log messages from the *STREAMS* log driver (`log(4)`) and writes these message to a log file. By default, `strerr(8)` logs all *STREAM* error messages from all drivers and modules.

  Messages that appear in the error log are intended to report exceptional conditions that require the attention of the person who administers your system.

  See `strerr(8)` for more information.

`strinfo(8)` (‘`/usr/sbin/strinfo`’)
: List *Stream* information. `strinfo(8)` is a rather useful *AIX* administrative utility that is also implemented by *Linux Fast-STREAMS*.

  The `strinfo(8)` C-language program can be used to list *Stream* instance information as well as information and statistics on a module or driver basis. The `scls(8)` utility provides a user program interface to the *STREAMS* Configuration module (`sc(4)`).

  This program is not even partially implemented in *Linux Fast-STREAMS* yet. User `proc(5)` file system and the ‘`/proc/streams`’ directory instead. Also, see `scls(8)` for driver and module specific information.

  See `strinfo(8)` for more information.

`strload(8)` (‘`/usr/sbin/strload`’)
: Loads the *STREAMS* subsystem. `strload(8)` is a useful *AIX* administrative utility that is also implemented by *Linux Fast-STREAMS*.

  The `strload(8)` bash script can be used to load *STREAMS* modules and drivers individually or from a configuration file.

  See `strload(8)` for more information.

`strsetup(8)` (‘`/usr/sbin/strsetup`’)
: Bash script.

  See `strsetup(8)` for more information.

`strvf(8)` (‘`/usr/sbin/strvf`’)
: C-language program.

  See `strvf(8)` for more information.

3.5.4 Performance Test Programs

Following are performance test programs:

`perftest(8)` (‘`/usr/sbin/perftest`’)
: C-language program.

  See `perftest(8)` for more information.

`perftestn(8)` (‘`/usr/sbin/perftestn`’)
: C-language program.

  See `perftestn(8)` for more information.

3.5.5 Conformance Test Programs

Following and conformance and validation testing programs:

`test-clone(8)` (‘`/usr/libexec/streams/test-clone`’)
: The `test-clone(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `clone(4)` *STREAMS* driver.

  See `test-clone(8)` for more information.

`test-connld(8)` (‘`/usr/libexec/streams/test-connld`’)
: The `test-connld(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `connld(4)` *STREAMS* driver.

  See `test-connld(8)` for more information.

`test-echo(8)` (‘`/usr/libexec/streams/test-echo`’)
: The `test-echo(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `echo(4)` *STREAMS* driver.

  See `test-echo(8)` for more information.

`test-fifo(8)` (‘`/usr/libexec/streams/test-fifo`’)
: The `test-fifo(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `fifo(4)` *STREAMS* driver.

  See `test-fifo(8)` for more information.

`test-log(8)` (‘`/usr/libexec/streams/test-log`’)
: The `test-log(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `log(4)` *STREAMS* driver.

  See `test-log(8)` for more information.

`test-loop(8)` (‘`/usr/libexec/streams/test-loop`’)
: The `test-loop(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `loop(4)` *STREAMS* driver.

  See `test-loop(8)` for more information.

`test-mux(8)` (‘`/usr/libexec/streams/test-mux`’)
: The `test-mux(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `mux(4)` *STREAMS* driver.

  See `test-mux(8)` for more information.

`test-nsdev(8)` (‘`/usr/libexec/streams/test-nsdev`’)
: The `test-nsdev(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `nsdev(4)` *STREAMS* driver.

  See `test-nsdev(8)` for more information.

`test-nuls(8)` (‘`/usr/libexec/streams/test-nuls`’)
: The `test-nuls(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `nuls(4)` *STREAMS* driver.

  See `test-nuls(8)` for more information.

`test-pipe(8)` (‘`/usr/libexec/streams/test-pipe`’)
: The `test-pipe(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `pipe(4)` *STREAMS* driver.

See `test-pipe(8)` for more information.

`test-pipemod(8)` (‘`/usr/libexec/streams/test-pipemod`’)
: The `test-pipemod(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `pipemod(4)` *STREAMS* driver.

  See `test-pipemod(8)` for more information.

`test-sad(8)` (‘`/usr/libexec/streams/test-sad`’)
: The `test-sad(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `sad(4)` *STREAMS* driver.

  See `test-sad(8)` for more information.

`test-sc(8)` (‘`/usr/libexec/streams/test-sc`’)
: The `test-sc(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `sc(4)` *STREAMS* driver.

  See `test-sc(8)` for more information.

`test-streams(8)` (‘`/usr/libexec/streams/test-streams`’)
: The `test-streams(8)` C-language program is a conformance and validation test program, in the OpenSS7 Project style, for the `STREAMS(9)` subsystem and primarily the `sth(4)` *Stream head*.

  See `test-streams(8)` for more information.

For the proper way to execute these validation test programs in a conformance and validation test suite, see Section 9.1.2.1 [Running Test Suites], page 137.

4 Development

For development using the *streams* package, See Section "About This Manual" in *STREAMS Programmer's Guide*.

4.1 Header Files

Header files are installed, typically, in the '`/usr/include/streams`' subdirectory. To use the header files from the package, '`-I/usr/include/streams`' must be included in the `gcc` command line as a compile option. This is true regardless of whether user space or kernel space programs are being compiled.

In general, '`-I`' include directives on the `gcc` command line should be ordered in the reverse order of the dependencies between packages. So, for example, if the include files from all add-on packages are required, the order of these directives would be: '`-I/usr/include/strss7 -I/usr/include/strsctp -I/usr/include/strinet -I/usr/include/strxnet -I/usr/include/strxns -I/usr/include/strcompat -I/usr/include/streams`'.

Following are the user visible header files provided by the `streams-0.9.2.4` package in directory '`/usr/include/streams`':

'`strlog.h`'
: This is the primary header file for the `strlog(4)` driver. It is normally only included by user space programs when interacting with the `log(4)` driver. See `log(4)` for more information.

'`stropts.h`'
: This is the primary user header file for the *Stream head*. It is normally only included by user space programs when interacting with the *Stream head*. See `sth(4)` for more information.

'`log.h`'
: This is the primary header file for the `log(4)` driver. It is normally only included by user space programs when interacting with the `log(4)` driver. See `log(4)` for more information.

'`loop.h`'
: This is the primary header file for the `loop(4)` driver. It is normally only included by user space programs when interacting with the `loop(4)` driver. See `loop(4)` for more information.

'`sad.h`'
: This is the primary header file for the `sad(4)` driver. It is normally only included by user space programs when interacting with the `sad(4)` driver. See `sad(4)` for more information.

'`sys/cmn_err.h`'
: This is the system specific kernel header file for the `cmn_err(9)` utility.

‘sys/ddi.h’
: This is the system specific kernel header file for various *STREAMS* DDI(9) utilities. It is normal only included by kernel space *STREAMS* modules and drivers. See DDI(9) for more information.

‘sys/debug.h’
: This is the system specific kernel header file for kernel debugging macros. It is normal only included by kernel space *STREAMS* modules and drivers.

‘sys/dki.h’
: This is the system specific kernel header file for various *STREAMS* DKI(9) utilities. It is normal only included by kernel space *STREAMS* modules and drivers. See DKI(9) for more information.

‘sys/kmem.h’
: This is the system specific kernel header file for kmem_alloc(9) and related utilities. It is normal only included by kernel space *STREAMS* modules and drivers. See kmem_alloc(9) for more information.

‘sys/strconf.h’
: This is the system specific kernel header file for *STREAMS* driver and module configuration. It is normal only included by kernel space *STREAMS* modules and drivers.

‘sys/strdebug.h’
: This is the system specific kernel header file for *STREAMS* driver and module debugging macros. It is normal only included by kernel space *STREAMS* modules and drivers.

‘sys/stream.h’
: This is the system specific kernel header file for *STREAMS* drivers and modules. It is normal only included by kernel space *STREAMS* modules and drivers. See STREAMS(9) for more information.

‘sys/strlog.h’
: This is the system specific header file for the strlog(4) and strlog(9) facilities. It is normally only included by kernel space programs when interacting with the log(4) driver. See log(4) for more information.

‘sys/stropts.h’
: This is the system specific user header file for the *Stream head*. It is normally only included by user space programs when interacting with the *Stream head*. See sth(4) for more information.

‘sys/stropts32.h’
: This is the system specific user 32/64-bit header file for the *Stream head*. It is normally only included by user space programs when interacting with the *Stream head*. See sth(4) for more information.

‘sys/strsubr.h’
: This is the system specific kernel header file for *STREAMS* private definitions. It is normal only included by kernel space *STREAMS* modules and drivers. See STREAMS(9) for more information.

‘sys/log.h’
: This is the system specific header file for the log(4) driver. It is normally only included by kernel space programs when interacting with the log(4) driver. See log(4) for more information.

‘sys/loop.h’
: This is the system specific header file for the loop(4) driver. It is normally only included by kernel space programs when interacting with the loop(4) driver. See loop(4) for more information.

‘sys/sad.h’
: This is the system specific header file for the sad(4) driver. It is normally only included by kernel space programs when interacting with the sad(4) driver. See sad(4) for more information.

‘sys/sc.h’
: This is the system specific header file for the sc(4) module. It is normally only included by user or kernel space programs when interacting with the sc(4) driver. See sc(4) for more information.

‘sys/testmod.h’
: This is the system specific header file for the testmod(4) module. It is normally only included by user or kernel space programs when interacting with the testmod(4) driver. See testmod(4) for more information.

4.1.1 User Space Programs

Typical include files for interacting with *STREAMS* from user space include the ‘stropts.h’ header file. Additional header files for interacting with specific drivers or modules may also be required.

4.1.2 Kernel Space Drivers and Modules

Typical include files for writing *STREAMS* modules and drivers for kernel space include the ‘sys/cmn_err.h’, ‘sys/kmem.h’, ‘sys/dki.h’, ‘sys/stream.h’, ‘sys/ddi.h’, and ‘sys/strconf.h’ header files. Additional header files for interacting with specific drivers or modules may also be required.

4.2 Libraries

Shared or static versions of the ‘libstreams’ library must be linked when using the streams-0.9.2.4 package. This library must either be specified on the gcc command line as a shared library (e.g. ‘-lstreams’) or as a static library (e.g. ‘/usr/lib/libstreams.a’).

If the shared library is linked, include the following options on the gcc command line:

‘-lstreams’
: Link to the ‘/usr/lib/libstreams.so’ shared library.

If the static library is linked, include the following options on the gcc command line:

‘/usr/lib/libstreams.a’
: Link to the ‘/usr/lib/libstreams.a’ static library.

4.3 Kernel Modules

Developing *STREAMS* kernel modules is similar to user space programs with regard to header files. '`/usr/include/streams`' should be placed as an include directory to search in the `gcc` command line. The rules for compiling *Linux* kernel modules should be followed. In particular, several important intricacies should be considered:

- The `gcc` compiler used to compile the kernel modules must be the same version of compiler that was used to compile the kernel.
- The `gcc` command line must have the same compile flags that were used to compile the kernel.
- The `gcc` command line must define several important kernel defines including '`-DLINUX`', '`-D__KERNEL__`', as well as the base name of the module.
- The `gcc` command line must include several important include files directly on the command line such as '`--include /lib/modules/2.4.20-28.7/build/include/linux/autoconf.h`' and maybe even '`--include /lib/modules/2.4.20-28.7/build/include/linux/modversions.h`'.[1]

4.4 Manual Pages

The `streams-0.9.2.4` package installs a number of manual pages in the '`/usr/share/man`' directory as follows:

The following manual pages are installed in Section 1 of the manual (in the subdirectory '`/usr/share/man/man1`'):

`strchg(1)` – change Stream configuration.
`strconf(1)` – query Stream configuration.
`strreset(1)` – reset a Stream.

The following manual pages are installed in Section 2 of the manual (in the subdirectory '`/usr/share/man/man2`'):

`fattach(2)` – name a *STREAMS* special file.
`fdetach(2)` – unname a *STREAMS* special file.
`getmsg(2)` – get next message off a Stream.
`getpmsg(2s)` – get next message off a Stream.
`isastream(2)` – test for a *STREAMS* special file.
`pipe(2s)` – create a *STREAMS* pipe.
`poll(2s)` – wait for an event on a *STREAMS* file descriptor.
`putmsg(2)` – put a message to a *STREAMS* character device.
`putpmsg(2s)` – put a band message to a *STREAMS* character device.
`read(2s)` – read from a file descriptor.
`readv(2s)` – read or write a vector.
`write(2s)` – write to a file descriptor.
`writev(2s)` – read or write a vector.

The following manual pages are installed in Section 3 of the manual (in the subdirectory '`/usr/share/man/man3`'):

[1] The kernel version '`2.4.20-28.7`' is just an example. For the running kernel, use the output of '`uname -r`'.

`streams(3)` – *STREAMS* system call library.
`libstreams(3)` – *STREAMS* system call library.
`LiS(3)` – *STREAMS* system call library.
`pLiS(3)` – *STREAMS* system call library.
`libLiS(3)` – *STREAMS* system call library.
`libpLiS(3)` – *STREAMS* system call library.
`fattach(3)` – name a *STREAMS* special file.
`fdetach(3)` – unname a *STREAMS* special file.
`isastream(3)` – test for a *STREAMS* special file.
`pipe(3)` – create a *STREAMS* pipe.
`pstrlog(3)` – print a *STREAMS* log buffer.
`s_pipe(3)` – create a *STREAMS* pipe.
`strlog(3)` – print a *STREAMS* log buffer.
`vstrlog(3)` – print a *STREAMS* log buffer.

The following manual pages are installed in Section 4 of the manual (in the subdirectory '`/usr/share/man/man4`'):

`bufmod(4)` – *STREAMS* buffering null module.
`clone(4)` – the *STREAMS* clone driver.
`connld(4)` – *STREAMS* connection line discipline module.
`conslog(4)` – *STREAMS* log device.
`echo(4)` – echo *STREAMS* device.
`fifo(4s)` – *STREAMS*-based FIFO device.
`log(4)` – *STREAMS* log device.
`loop(4)` – *STREAMS* loop-around pseudo-device driver.
`loop_clone(4)` – *STREAMS* loop-around pseudo-device driver.
`mux(4)` – *STREAMS* multiplexing pseudo-device driver.
`nsdev(4)` – named *STREAMS* device.
`nullmod(4)` – *STREAMS* null module.
`nuls(4)` – null *STREAMS* device.
`pipe(4)` – *STREAMS* bi-directional pipe device.
`pipemod(4)` – *STREAMS*-based pipe module.
`s_fifo(4)` – *STREAMS*-based FIFO device.
`sad(4)` – *STREAMS* Administrative Driver.
`sc(4)` – *STREAMS* Configuration module.
`sfx(4)` – *STREAMS*-based FIFO device.
`sloop(4)` – *STREAMS* loop-around pseudo-device driver.
`spx(4)` – *STREAMS* bi-directional pipe device.
`sth(4)` – *STREAMS* Stream head module.
`strlog(4)` – *STREAMS* log device.
`testmod(4)` – *STREAMS* test module.

The following manual pages are installed in Section 5 of the manual (in the subdirectory '`/usr/share/man/man5`'):

`autopush(5)` – control the autopush module list for a *STREAMS* device.
`specfs(5)` – *STREAMS* special device shadow file system.
`strapush(5)` – *STREAMS* autopush structure.

`strioctl(5)` – *STREAMS* I/O control data structure.
`strsetup.conf(5)` – configuration file for *STREAMS* drivers.

The following manual pages are installed in Section 7 of the manual (in the subdirectory '`/usr/share/man/man7`'):

`streamio(7)` – *STREAMS* ioctl commands.
`I_ANCHOR(7)` – *STREAMS* anchor input-output control.
`I_ATMARK(7)` – check if a *STREAMS* message is marked.
`I_CANPUT(7)` – check if a *STREAMS* band is writable.
`I_CKBAND(7)` – check if a *STREAMS* band is readable.
`I_EGETSIG(7)` – get enhanced *STREAMS* SIGPOLL events.
`I_ESETSIG(7)` – set enhanced *STREAMS* SIGPOLL events.
`I_FATTACH(7)` – emulate `fattach(2)` system call.
`I_FDETACH(7)` – emulate `fdetach(2)` system call.
`I_FDINSERT(7)` – insert a Stream identifier into a *STREAMS* message and send it downstream.
`I_FIND(7)` – find a *STREAMS* module on a Stream.
`I_FLUSH(7)` – flush messages from a *STREAMS* special file.
`I_FLUSHBAND(7)` – flush messages for a band from a *STREAMS* special file.
`I_GERROPT(7)` – get error options for a *STREAMS* file.
`I_GETBAND(7)` – get band number of a message on a Stream.
`I_GETCLTIME(7)` – get close time for a *STREAMS* file.
`I_GETPMSG(7)` – *STREAMS* `getpmsg(2s)` system call emulation.
`I_GETSIG(7)` – get SIGPOLL events.
`I_GRDOPT(7)` – get *STREAMS* read options.
`I_GWROPT(7)` – get *STREAMS* write options.
`I_ISASTREAM(7)` – emulate `isastream(2)` system call.
`I_LINK(7)` – link a Stream beneath a *STREAMS* multiplexing driver.
`I_LIST(7)` – list *STREAMS* module names on a Stream.
`I_LOOK(7)` – look at topmost *STREAMS* module on a Stream.
`I_NREAD(7)` – number of unread bytes on a Stream.
`I_PEEK(7)` – peek at *STREAMS* message on read queue.
`I_PIPE(7)` – obtain a *STREAMS* based pipe.
`I_PLINK(7)` – persistently link a Stream beneath a *STREAMS* multiplexing driver.
`I_POP(7)` – pop a *STREAMS* module from a Stream.
`I_PUNLINK(7)` – unlink a *STREAMS* persistent link.
`I_PUSH(7)` – push a *STREAMS* module on a Stream.
`I_PUTPMSG(7)` – *STREAMS* `putpmsg(2s)` system call emulation.
`I_RECVFD(7)` – receive a file descriptor on a Stream.
`I_SENDFD(7)` – send a file descriptor on a Stream.
`I_SERROPT(7)` – set error options for a *STREAMS* file.
`I_SETCLTIME(7)` – set close time for a *STREAMS* file.
`I_SETSIG(7)` – set SIGPOLL events.
`I_SRDOPT(7)` – set *STREAMS* read options.

`I_STR(7)` – *STREAMS* intput-output control.
`I_SWROPT(7)` – set *STREAMS* write options.
`I_UNLINK(7)` – unlink a Stream from a *STREAMS* multiplexing driver.

The following manual pages are installed in Section 8 of the manual (in the subdirectory '`/usr/share/man/man8`'):

`autopush(8)` – control the autopush module list for a *STREAMS* device.
`fattach(8)` – name a *STREAMS* file.
`fdetach(8)` – unname a *STREAMS* file.
`insf(8)` – install special device files.
`perftest(8)` – *STREAMS* benchmark performance tests on a pipe.
`perftestn(8)` – *STREAMS* benchmark performance tests on a pipe.
`scls(8)` – produce a list of *STREAMS* module and driver names.
`specfs(8)` – System V Init Script for the *STREAMS* Shadow Special Filesystem.
`specfs.sh(8)` – System V Init Script for the *STREAMS* Shadow Special Filesystem.
`strace(8)` – write *STREAMS* event trace messages to the standard output.
`strclean(8)` – clean up the *STREAMS* error logger.
`streams(8)` – System V Init Script for the *STREAMS* subsystem.
`streams_mknod(8)` – create or remove *STREAMS* device nodes.
`streams.sh(8)` – System V Init Script for the *STREAMS* subsystem.
`strerr(8)` – receive error log messages from the *STREAMS* `log(4)` driver.
`strinfo(8)` – display information about *STREAMS* devices.
`strload(8)` – load the *STREAMS* subsystem.
`strsetup(8)` – *STREAMS* setup command.
`strvf(8)` – *STREAMS* verification tool.
`test-clone(8)` – a test suite executable for the `clone(4)` *STREAMS* driver.
`test-connld(8)` – a test suite executable for the `connld(4)` *STREAMS* module.
`test-echo(8)` – a test suite executable for the `echo(4)` *STREAMS* driver.
`test-fifo(8)` – a test suite executable for the `fifo(4)` *STREAMS* driver.
`test-log(8)` – a test suite executable for the `log(4)` *STREAMS* driver.
`test-loop(8)` – a test suite executable for the `loop(4)` *STREAMS* driver.
`test-mux(8)` – a test suite executable for the `mux(4)` *STREAMS* driver.

`test-nsdev(8)` – a test suite executable for the `nsdev(4)` *STREAMS* driver.

`test-nuls(8)` – a test suite executable for the `nuls(4)` *STREAMS* driver.

`test-pipe(8)` – a test suite executable for the `pipe(4)` *STREAMS* driver.

`test-pipemod(8)` – a test suite executable for the `pipemod(4)` *STREAMS* module.

`test-sad(8)` – a test suite executable for the `sad(4)` *STREAMS* driver.

`test-sc(8)` – a test suite executable for the `sc(4)` *STREAMS* module.

`test-streams(8)` – a test suite executable for *STREAMS*.

The following manual pages are installed in Section 9 of the manual (in the subdirectory '`/usr/share/man/man9`'):

`Intro(9)` – introduction to *STREAMS* kernel functions.

`STREAMS(9)` – introduction to *STREAMS* kernel functions.

`SPG(9)` – *Linux Fast-STREAMS* Programmers Guide.

`DDI(9)` – Device Driver interface/Driver Kernel Interface.

`LfS(9)` – introduction to *STREAMS* kernel functions.

`mp-streams(9)` – multi-processor *STREAMS* executive.

`M_BACKDONE(9)` – *STREAMS* backwash done direct I/O message.

`M_BACKWASH(9)` – *STREAMS* backwash direct I/O message.

`M_BREAK(9)` – *STREAMS* break message.

`M_COPYIN(9)` – *STREAMS* copyin message.

`M_COPYOUT(9)` – *STREAMS* copyout message.

`M_CTL(9)` – *STREAMS* control message.

`M_DATA(9)` – *STREAMS* data message.

`M_DELAY(9)` – *STREAMS* delay message.

`M_DONTPLAY(9)` – *STREAMS* don't play direct I/O message.

`M_ERROR(9)` – *STREAMS* error message.

`M_EVENT(9)` – *STREAMS* event message.

`M_FLUSH(9)` – *STREAMS* flush message.

`M_HANGUP(9)` – *STREAMS* hangup message.

`M_HPDATA(9)` – *STREAMS* high priority data message.

`M_IOCACK(9)` – *STREAMS* IO control acknowledgement message.

`M_IOCDATA(9)` – *STREAMS* IO control data message.

`M_IOCNAK(9)` – *STREAMS* IO control negative acknowledgement message.

`M_IOCTL(9)` – *STREAMS* IO control message.

`M_LETSPLAY(9)` – *STREAMS* let's plan direct I/O message.

`M_NOTIFY(9)` – *STREAMS* notify message.

`M_PASSFP(9)` – *STREAMS* pass file pointer message.

`M_PCCTL(9)` – *STREAMS* priority control message.

`M_PCEVENT(9)` – *STREAMS* priority event message.

`M_PCPROTO(9)` – *STREAMS* priority protocol message.

M_PCRSE(9) – *STREAMS* priority reserved message.
M_PCSETOPTS(9) – *STREAMS* priority set options message.
M_PCSIG(9) – *STREAMS* priority signal message.
M_PROTO(9) – *STREAMS* protocol message.
M_READ(9) – *STREAMS* read message.
M_RSE(9) – *STREAMS* reserved message.
M_SETOPTS(9) – *STREAMS* set options message.
M_SIG(9) – *STREAMS* signal message.
M_START(9) – *STREAMS* start message.
M_STARTI(9) – *STREAMS* start input message.
M_STOP(9) – *STREAMS* stop message.
M_STOPI(9) – *STREAMS* stop input message.
M_TRAIL(9) – *STREAMS* trail message.
M_UNHANGUP(9) – *STREAMS* unhangup message.
OTHERQ(9) – return other queue of a *STREAMS* queue pair.
QNORM(9) – *STREAMS* data block structure.
QPCTL(9) – *STREAMS* data block structure.
RD(9) – return the read queue of a *STREAMS* queue pair.
SAMESTR(9) – test for a *STREAMS* pipe or FIFO.
WR(9) – return the write queue of a *STREAMS* queue pair.
adjmsg(9) – trim bytes from the front or back of a *STREAMS* message.
allocb(9) – allocate a *STREAMS* message and data block.
alloclk(9) – allocate or free a *STREAMS* link block.
allocq(9) – allocate a *STREAMS* queue pair.
allocstr(9) – allocate a *STREAMS* Stream head.
appq(9) – append one *STREAMS* message after another.
apush_get(9) – get the autopush list associated with a *STREAMS* driver.
apush_set(9) – set the autopush list associated with a *STREAMS* driver.
apush_vml(9) – verify a *STREAMS* module list.
autopush(9) – perform autopush operations on a newly opened Stream.
autopush_add(9) – add an autopush list entry for a given *STREAMS* device number.
autopush_del(9) – delete an autopush list entry for a given *STREAMS* device number.
autopush_find(9) – find an autopush list entry for a given *STREAMS* device number.
autopush_search(9) – find an autopush list entry for a given *STREAMS* device name and number.
autopush_vml(9) – verify a *STREAMS* module list.
backq(9) – find the upstream or downstream queue.
bcanget(9) – test for message arrival on a band on a Stream.
bcangetany(9) – check whether messages are in any (non-zero) band.

bcanput(9) – test flow control on a *STREAMS* message queue.
bcanputany(9) – check if a message can be put to any (non-zero) band on a queue.
bcanputnext(9) – test flow control on the next *STREAMS* message queue.
bcanputnextany(9) – check if a message can be put to any (non-zero) band on the next queue.
bcid_t(9) – install a buffer callback.
bcmp(9) – compare byte strings.
bcopy(9) – copy byte strings.
bufcall(9) – install a buffer callback.
bufcall_id_t(9) – install a buffer callback.
bzero(9) – zero a byte string.
canenable(9) – test whether a *STREAMS* message queue can be scheduled.
canget(9) – test for message arrival on a Stream.
canput(9) – test flow control on a *STREAMS* message queue.
canputnext(9) – test flow control on the next *STREAMS* message queue.
cdev_count(9) – character device switch table helper functions.
cdev_find(9) – character device switch table helper functions.
cdev_match(9) – character device switch table helper functions.
cdev_minor(9) – character device switch table helper functions.
cdev_str(9) – character device switch table helper functions.
cdevsw(9) – the *SVR 4* character device switch table structure.
cdevsw_list(9) – the *SVR 4* character device switch table structure.
cdevsw_lock(9) – the *SVR 4* character device switch table structure.
cdrv_get(9) – the *SVR 4* character device switch table structure.
cdrv_put(9) – the *SVR 4* character device switch table structure.
cmaj_add(9) – major/minor character device node helper functions.
cmaj_del(9) – major/minor character device node helper functions.
cmaj_get(9) – major/minor character device node helper functions.
cmin_add(9) – major/minor character device node helper functions.
cmin_count(9) – major/minor character device node helper functions.
cmin_del(9) – major/minor character device node helper functions.
cmin_find(9) – major/minor character device node helper functions.
cmin_get(9) – major/minor character device node helper functions.
cmin_ini(9) – major/minor character device node helper functions.
cmin_rel(9) – major/minor character device node helper functions.
cmn_err(9) – print a kernel command error.
copyb(9) – copy a *STREAMS* message block.
copyin(9) – copy user data in from user space to kernel space.
copymsg(9) – copy a *STREAMS* message.
copyout(9) – copy user data in from kernel space to user space.
copyreq(9) – *STREAMS* copy request block structure.
copyresp(9) – *STREAMS* copy response block structure.

`cred_t(9)` – credentials structure.
`ctlmsg(9)` – test a *STREAMS* message type for control.
`datab(9)` – *STREAMS* data block structure.
`datamsg(9)` – test a *STREAMS* message type for data.
`dblk_t(9)` – *STREAMS* data block structure.
`delay(9)` – postpone the calling process for a number of clock ticks.
`dev_t(9)` – *STREAMS* device type.
`devnode(9)` – *STREAMS* character device node structure.
`do_fattach(9)` – implement the `fattach(2)` system call.
`do_fdetach(9)` – implement the `fdetach(2)` system call.
`do_spipe(9)` – implement the `pipe(2s)` system call.
`drv_getparm(9)` – driver retrieve kernel parameter.
`drv_hztomsec(9)` – convert kernel tick time between microseconds or milliseconds.
`drv_hztousec(9)` – convert kernel tick time between microseconds or milliseconds.
`drv_msectohz(9)` – convert kernel tick time between microseconds or milliseconds.
`drv_priv(9)` – check if current process is privileged.
`drv_usectohz(9)` – convert kernel tick time between microseconds or milliseconds.
`drv_usecwait(9)` – delay for a number of microseconds.
`dupb(9)` – duplicate a *STREAMS* message block.
`dupmsg(9)` – duplicate a *STREAMS* message.
`enableok(9)` – allow a *STREAMS* message queue to be scheduled.
`enableq(9)` – schedule a *STREAMS* message queue service procedure.
`esballoc(9)` – allocate a *STREAMS* message and data block with caller supplied data buffer.
`esbbcall(9)` – install a buffer callback for an extended *STREAMS* message block.
`flushband(9)` – flushes a band of *STREAMS* messages from a queue.
`flushq(9)` – flush messages from a *STERAMS* message queue.
`fmod_add(9)` – file module switch table helper functions.
`fmod_count(9)` – file module switch table helper functions.
`fmod_del(9)` – file module switch table helper functions.
`fmod_find(9)` – file module switch table helper functions.
`fmod_get(9)` – file module switch table helper functions.
`fmod_put(9)` – file module switch table helper functions.
`fmod_str(9)` – file module switch table helper functions.
`fmodsw(9)` – the *SVR 4 STREAMS* module switch table.
`fmodsw_list(9)` – the *SVR 4 STREAMS* module switch table.
`fmodsw_lock(9)` – the *SVR 4 STREAMS* module switch table.
`freeb(9)` – frees a *STREAMS* message block.
`freelk(9)` – allocate or free a *STREAMS* link block.

freemsg(9) – frees a *STREAMS* message.
freeq(9) – deallocate a *STREAMS* queue pair.
freestr(9) – deallocate a *STREAMS* Stream head.
freezestr(9) – freeze the state of a Stream.
frtn_t(9) – allocate a *STREAMS* message and data block with caller supplied data buffer.
getadmin(9) – get the administrative function pointer for a *STREAMS* module.
getmajor(9) – get the internal major device number for a device.
getmid(9) – get the *STREAMS* module id for a name.
getminor(9) – get the extended minor device number for a device.
getq(9) – get a message from a *STREAMS* message queue.
insq(9) – insert a message into a *STREAMS* message queue.
iocblk(9) – *STREAMS* input-output control block structure.
isdatablk(9) – test a *STREAMS* data block for data type.
isdatamsg(9) – test a *STREAMS* data block for data type.
kmem_alloc(9) – allocate kernel memory.
kmem_alloc_node(9) – allocate kernel memory.
kmem_free(9) – deallocate kernel memory.
kmem_zalloc(9) – allocate and zero kernel memory.
kmem_zalloc_node(9) – allocate and zero kernel memory.
linkb(9) – link a message block to a *STREAMS* message.
linkblk(9) – *STREAMS* link block structure.
linkmsg(9) – link a message block to a *STREAMS* message.
major_t(9) – get the internal major device number for a device.
makedevice(9) – create a device from major and minor device numbers.
max(9) – determine the maximum of two integers.
mblk_t(9) – *STREAMS* message block structure.
min(9) – determine the minimum of two integers.
minor_t(9) – get the extended minor device number for a device.
modID_t(9) – get the *STREAMS* module id for a name.
module_info(9) – *STREAMS* module information structure.
module_stat(9) – *STREAMS* module statistics structure.
module_stat_t(9) – *STREAMS* module statistics structure.
msgb(9) – *STREAMS* message block structure.
msgdsize(9) – calculate the size of the data in a *STREAMS* message.
msgpullup(9) – pull up bytes in a *STREAMS* message.
msgsize(9) – calculate the size of the message blocks in a *STREAMS* message.
noenable(9) – disable a *STREAMS* message queue from being scheduled.
pcmsg(9) – test a data block message type for priority control.
pullupmsg(9) – pull up the bytes in a *STREAMS* message.

put(9) – invoke the put procedure for a *STREAMS* driver of module with a *STREAMS* message.

putbq(9) – put a message back on a *STREAMS* message queue.

putctl(9) – put a control message on a *STREAMS* message queue.

putctl1(9) – put a one-byte control message on a *STREAMS* message queue.

putctl2(9) – put a two-byte control message on a *STREAMS* message queue.

putnext(9) – put a message on the downstream *STREAMS* message queue.

putnextctl(9) – put a control message on the next *STREAMS* message queue.

putnextctl1(9) – put a one-byte control message on the next *STREAMS* message queue.

putnextctl2(9) – put a two-byte control message on the next *STREAMS* message queue.

putq(9) – put a message on a *STREAMS* message queue.

qattach(9) – attach a module onto a *STREAMS* file.

qbackenable(9) – perform back enabling on a *STREAMS* queue.

qband(9) – queue band structure.

qband_t(9) – queue band structure.

qclose(9) – close a *STREAMS* driver or module.

qcountstrm(9) – add all counts on all *STREAMS* message queues in a Stream.

qdelete(9) – delete a queue pair from a Stream.

qdetach(9) – detach a module from a *STREAMS* file.

qenable(9) – schedule a *STREAMS* message queue service procedure.

qfields(9) – set attributes of a *STREAMS* message queue.

qfields_t(9) – set attributes of a *STREAMS* message queue.

qi_putp(9) – *STREAMS* driver or module put procedure.

qi_putp_t(9) – *STREAMS* driver or module put procedure.

qi_qadmin(9) – *STREAMS* driver or module admin routine.

qi_qadmin_t(9) – *STREAMS* driver or module admin routine.

qi_qclose(9) – *STREAMS* driver or module close routine.

qi_qclose_t(9) – *STREAMS* driver or module close routine.

qi_qopen(9) – *STREAMS* driver or module open routine.

qi_qopen_t(9) – *STREAMS* driver or module open routine.

qi_srvp(9) – *STREAMS* driver or module service procedure.

qi_srvp_t(9) – *STREAMS* driver or module service procedure.

qinit(9) – *STREAMS* queue initialization structure.

qinsert(9) – insert a queue pair beneath another queue pair in a Stream.

qopen(9) – call a *STREAMS* driver or module open routine.

qprocsoff(9) – disable *STREAMS* message queue processing for multi-processing.

qprocson(9) – enable a *STREAMS* message queue for multi-processing.

qready(9) – test if queue procedures are scheduled.

qreply(9) – replies to a message from a *STREAMS* message queue.

qscan(9) – place a queue on the scan list.

qsize(9) – return the number of messages on a queue.

queue(9) – *STREAMS* message queue structure.

queue_t(9) – *STREAMS* message queue structure.

register_clone(9) – register a clone(4) minor.

register_cmajor(9) – register external device major number.

register_ioctl32(9) – register a 32-bit IO control command.

register_strdev(9) – register a *STREAMS* device.

register_strdrv(9) – register a *STREAMS* driver.

register_strlog(9) – register a *STREAMS* logger.

register_strmod(9) – register a *STREAMS* module.

register_strnod(9) – register a *STREAMS* minor device node.

rmvb(9) – remove a message block from a *STREAMS* message.

rmvq(9) – remove a message from a *STREAMS* message queue.

runqueues(9) – run queue service procedures and other asynchronous *STREAMS* events.

sd_get(9) – acquire and release a reference to the Stream head.

sd_put(9) – acquire and release a reference to the Stream head.

sdev_add(9) – character device switch table helper functions.

sdev_del(9) – character device switch table helper functions.

sdev_get(9) – character device switch table helper functions.

sdev_ini(9) – character device switch table helper functions.

sdev_put(9) – character device switch table helper functions.

sdev_rel(9) – character device switch table helper functions.

sealloc(9) – *STREAMS* event allocators.

sefree(9) – *STREAMS* event allocators.

setq(9) – set sizes and procedures associated with a *STREAMS* message queue.

setqsched(9) – invoke the *STREAMS* scheduler.

setsq(9) – set synchronization queues, sizes and procedures associated with a *STREAMS* message queue.

skballoc(9) – allocate a *STREAMS* message and data block with a caller supplied socket buffer.

spec_open(9) – *STREAMS* special device shadow file system.

spec_reparent(9) – *STREAMS* special device shadow file system.

specfs_mount(9) – *STREAMS* special device shadow file system.

specfs_umount(9) – *STREAMS* special device shadow file system.

str_close(9) – Stream head module procedures.

str_open(9) – Stream head module procedures.

streamtab(9) – *STREAMS* module definition structure.
streamtab_t(9) – *STREAMS* module definition structure.
strgetpmsg(9) – perform a getpmsg(2s) operation on a Stream head.
strioctl(9) – perform a ioctl(2s) operation on a Stream head.
strlog(9) – pass a message to the *STREAMS* logger.
strm_f_ops(9) – file operations for Stream heads.
stroptions(9) – *STREAMS* Stream head options structure.
strpoll(9) – perform a poll(2s) operation on a Stream head.
strputpmsg(9) – perform a putpmsg(2s) operation on a Stream head.
strqget(9) – get attributes of a *STREAMS* message queue.
strqset(9) – set attributes of a *STREAMS* message queue.
strread(9) – perform a read(2s) operation on a Stream head.
strrput(9) – Stream head module procedures.
strsendpage(9) – perform a sendfile(2s) operation on a Stream head.
strthread(9) – the *SVR 4 STREAMS* scheduler thread structure and array.
strthreads(9) – the *SVR 4 STREAMS* scheduler thread structure and array.
strwput(9) – Stream head module procedures.
strwrite(9) – perform a write(2s) operation on a Stream head.
strwsrv(9) – Stream head module procedures.
sysctl_str_nstrpush(9) – introduction to *STREAMS* kernel functions.
sysctl_str_strctlsz(9) – introduction to *STREAMS* kernel functions.
sysctl_str_strmsgsz(9) – introduction to *STREAMS* kernel functions.
testb(9) – test if a *STREAMS* message can be allocated.
timeout(9) – start a timer.
timeout_id_t(9) – start a timer.
timo_fcn_t(9) – start a timer.
toid_t(9) – start a timer.
unbufcall(9) – remove a *STREAMS* buffer callback.
unfreezestr(9) – thaw the state of a Stream queue.
unlinkb(9) – unlink a message block from a *STREAMS* message.
unlinkmsg(9) – unlink a message block from a *STREAMS* message.
unregister_clone(9) – unregister a clone(4) minor.
unregister_cmajor(9) – unregister external device major number.
unregister_ioctl32(9) – unregister a 32-bit IO control command.
unregister_strdev(9) – unregister a *STREAMS* device.
unregister_strdrv(9) – unregister a *STREAMS* driver.
unregister_strmod(9) – unregister a *STREAMS* module.
unregister_strnod(9) – unregister a *STREAMS* minor device node.
untimeout(9) – stop a timer.
unweldq(9) – unweld two queues.
vcmn_err(9) – print a kernel command error.
vstrlog(9) – pass a message to the *STREAMS* logger.
vstrlog_t(9) – register a new *STREAMS* log device.

`weld_arg_t(9)` – weld two (or four) queues together.
`weld_fcn_t(9)` – weld two (or four) queues together.
`weldq(9)` – weld two (or four) queues together.
`xmsgsize(9)` – calculate the size of message blocks in a *STREAMS* message.

5 Porting

Linux Fast-STREAMS provides a rich set of *STREAMS* functions, DDI/DKI functions and utilities based on *SVR 4.2 MP* for the development of *STREAMS* modules and drivers. Although these functions and capabilities provide all of the utilities necessary for the development of *STREAMS* modules and drivers, it represents the common set of functions provided by other *STREAMS* implementations.

Some other *STREAMS* implementations provide interfaces, utilities and helper functions specific to those implementations. Where *STREAMS* implementations differ the most is in the manner in which they configure and register *STREAMS* drivers and modules for interface to the operating system, including registration functions, device numbering, creation of minor device nodes, administration and other mechanisms not specified by the *System V Release 4 Programmer's Guide – STREAMS*.

To assist with porting of *STREAMS* drivers and modules from other *STREAMS* implementations and *UNIX* based operating systems to *Linux Fast-STREAMS*, *Linux Fast-STREAMS* provides a separate *STREAMS Compatibility* add-on package, called '`strcompat-0.9.2.7`',[1] that provide source level compatibility with a wide range of mainstream *STREAMS* implementations and significant groups of compatibility and helper functions (such as those from *Solaris* and *Mentat*). These compatibility packages also provide separate demand loadable kernel modules that provide the additional compatibility functionality with *Linux Fast-STREAMS*.

Perhaps one of the most important ports to *Linux Fast-STREAMS* is from the deprecated and deficient *LiS* package to *Linux Fast-STREAMS*. The *STREAMS Compatibility* package, '`strcompat`', also provides a source level compatibility module that provides source level compatibility to *LiS* to ease porting *LiS* drivers and modules to the superior *Linux Fast-STREAMS*.

In general, when porting to *Linux Fast-STREAMS* from another *STREAMS* implementation, the following items will need the most attention:

Header Files
: The *STREAMS* and operating system specific header files that must be included by kernel modules to implement *STREAMS* drivers or modules are specific to each *STREAMS* implementation. Although there are some basic header files to include ('`sys/stream.h`', '`sys/strconf.h`', '`sys/ddi.h`', *sys/cmn_err.h*, *sys/dki.h*, *sys/kmem.h*), the order in which these headers are included and the additional operating system specific headers are implementation specific. See the example drivers and modules for the header files that are necessary for *Linux Fast-STREAMS STREAMS* modules and drivers.

Kernel Module Mechanism
: The mechanism for creating, configuring and loading kernel modules is specific to the operating system implementation. *Linux Fast-STREAMS* uses the normal *Linux* mechanisms for kernel modules also for *STREAMS* drivers and modules.

[1] Previously the *STREAMS Compatibility* modules were part of the *Linux Fast-STREAMS* base package. They were separated to an add-on package for a number of reasons that are described in the *OpenSS7 STREAMS Compatibility – Installation and Reference Manual*.

Configuration and Registration
: The *STREAMS* driver or module will need to be converted to use the *Linux Fast-STREAMS* configuration and registration mechanisms. See `register_strdev(9)`, `unregister_strdev(9)`, `register_strmod(9)` and `unregister_strmod(9)` for more specific information on the *Linux Fast-STREAMS* configuration and registration mechanisms.

Non-STREAMS DDI/DKI Facilities
: Any of the non-*STREAMS* DDI/DKI facilities or operating system specific facilities that are used by the *STREAMS* driver or module may need to be replaced with the *Linux* equivalent. Examples of such facilities include basic locks, read-write locks, semaphores and mutexes, atomic integers, interrupt suppression, bus access and memory mapping functions.

Binary Modules
: When *STREAMS* drivers or modules are released as binary objects and source code is not available, it is still possible to convert the binary module for use with *Linux Fast-STREAMS*. The facility to convert binary modules for use with *Linux Fast-STREAMS* is not, however, part of the base package and is not part of the *STREAMS Compatibility* package. A separate add-on package, the *Binary Compatibility Modules* package, '`strbcm-0.9.2.5`' was developed explicitly for this purpose.[2]

5.1 Porting from LiS

Applications programs to not need to be ported, or even recompiled when they use shared libraries. *Linux Fast-STREAMS* provides *LiS* compatible shared object libraries. Applications compiled against static libraries will need to be recompiled unless *Linux Fast-STREAMS* was configured for *STREAMS* binary compatibility mode.[3]

In general, if no *LiS* specific functions are used (other than *STREAMS* driver or module registration functions), porting of *LiS* drivers and modules to *Linux Fast-STREAMS* is straightforward. *Linux Fast-STREAMS* provides several *STREAMS* drivers and modules that are common to both OpenSS7 *Linux Fast-STREAMS* and *LiS* releases. These drivers and modules provide examples of how to write *STREAMS* drivers or modules that can run under either *LiS* or *Linux Fast-STREAMS*. The common modules and drivers are as follows:

'`src/drivers/echo.c`'	`echo(4)`
'`src/drivers/mux.c`'	`mux(4)`
'`src/drivers/nuls.c`'	`nuls(4)`
'`src/modules/nullmod.c`'	`nullmod(4)`
'`src/modules/sc.c`'	`sc(4)`
'`src/modules/testmod.c`'	`testmod(4)`

When built for *Linux Fast-STREAMS*, '`C`' preprocessor symbol '`LFS`' is defined; when built for *LiS*, '`LIS`' is defined.

---

[2] Because binary compatibility is a thorny issue, both from the standpoint of technical merit and licensing issues, the '`strbcm`' package is only currently available to subscribers and sponsors of the *OpenSS7 Project*.

[3] See the '`--enable-streams-bcm`' option to `configure` in Section 8.3.5.1 [Configure Options], page 104.

LiS provides many simple wrapper functions that are *Linux* kernel functions with '`lis_`' prepended to the name. Aside from licensing issues associated with using these wrapper functions, in many cases it is possible to simply drop the '`lis_`' from the function call and use the *Linux* functions directly. This is true for most spin locks, read-write locks, semaphores, mutexes, atomic integers, bus access and memory mapping functions.

When many or specific *LiS* functions calls are necessary, it is better to use the *LiS* compatibility module present in the '`strcompat-0.9.2.7`' package.

5.2 Porting from SVR 4.2 MP

When porting from *SVR 4.2 MP* or a *STREAMS* implementation based closely on *SVR 4.2 MP*, such as *SUPER-UX*, *UXP/V*, *IRIX* or many of the real-time operating system implementations (e.g. *VxWorks*), it is possible to port directly to *Linux Fast-STREAMS* without using the *STREAMS Compatibility* package. Event when porting from *AIX*, *HP-UX* and *OSF/1* it is possible to avoid using the compatibility package.

Most pseudo-device drivers and modules should not require any special facilities beyond basic locks and porting may be straightforward. Where extensive implementation specific DDI/DKI or operating system functions are required, it is better to use the *STREAMS Compatibility* package and modules closest to the specific implementation being ported from.

5.3 Porting from Solaris

When porting from *Solaris* there are both *STREAMS* facilities and extensive DDI/DKI facilities that differ greatly from basic *SVR 4.2 MP STREAMS* and DDI/DKI functions. For porting all but the most trivial of *STREAMS* drivers and modules written specifically for *Solaris*, it is better to use the *STREAMS Compatibility* package and the *Solaris* compatibility module provided by that package.[4]

5.4 Porting from UnixWare

When porting from *UnixWare* there are extensive operating system facilities that differ greatly from basic *Linux* facilities. For the most part these are basic locks, read-write locks, condition variables, sleep locks, atomic integers, bus access and mapping functions. Although *Linux* provides equivalents in most of these categories, the *STREAMS Compatibility* package contains a compatibility module for *UnixWare* that provides source compatibility with most of these functions. It is recommended that all but the most trivial of *UnixWare* drivers and modules use the *STREAMS Compatibility* package when porting.

5.5 Porting from Mentat

When porting a *STREAMS* driver or module from a *Mentat* implementation (such as *AIX*, *HP-UX*, *OSF/1*, *Mac OT*) that makes heavy use of the *Mentat* '`mi_`' or '`mps_`' helper functions, it is best to use the OpenSS7 implementations of those functions available in the *STREAMS Compatibility* package directly. The *STREAMS Compatibility* package

[4] The compatibility package also provides a version of the '`strsun.h`' header file and the helper functions provided there.

provides a *Mentat Portable STREAMS* compatibility module that provides implementations of the *Mentat* functions found in *AIX*, *OSF/1* and *Mac OT*.[5]

[5] Note that *Solaris* also provides versions of the *Mentat* functions which appear in the *Solaris* compatiblity module.

6 Conformance

6.1 Standards Compliance

Linux Fast-STREAMS was designed and implemented to be compliant with as many standards impinging on *STREAMS* as possible. There are three areas of standards compliance as follows:

6.1.1 User Interface Compliance

The *STREAMS* user interface standards are primarily specified by the *IEEE* and *OpenGroup* standards and take the form of the *POSIX 2003* and *Single UNIX Specification* standards simultaneously released by the *OpenGroup* in conjunction with *IEEE*. The latest *POSIX/IEEE/OpenGroup* standard provide an *XSI* extension that includes the *STREAMS* user interface. For the most part, the *OpenGroup* XSI interface is completely compatible with the user interface described in the *System V Release 4 Programmer's Manual – STREAMS*, and where it does not, Stream head options are provided to select between the default *OpenGroup* XSI behaviour and the traditional *SVR 4* behaviour.

Most of the XSI specifications of the *OpenGroup* describe the behaviour of the Stream head and the behaviour of specific *STREAMS* drivers or modules (such as pipes, FIFOs and terminals). Also described is the `poll(2s)` behaviour, generation of signals, and `read(2s)` and `write(2s)` behaviour as it applies to *STREAMS* character special devices.[1]

User interface compliance of the *Linux Fast-STREAMS* is tested with custom validation test suites that ship with the package. See Section 3.5.5 [Conformance Test Programs], page 35 for more information on conformance and validation test suites.

6.1.2 Service Interface Compliance

The *OpenGroup* (now and in previous incarnations) have issued standardized service interface specifications as part of the Common Application Environment (CAE) specifications. These service interface specifications usually concern networking interfaces such as the *Data Link Provider Interface (DLPI)*, the *Network Provider Interface (NPI)*, the *Transport Provider Interface (TPI)*, the *X/Open Transport Interface (XTI)* and the *Sockets* API. Although these standards impinge upon various networking add-on packages for *Linux Fast-STREAMS*, they do not impinge upon the base *STREAMS* package documented here. See the *Installation and Reference Manual* for the appropriate add-on package.

6.1.3 Kernel Interface Compliance

The *STREAMS* kernel interfaces, DDI/DKI and other facilities available to the *STREAMS* driver or module writer has not been subjected to formal standardization. For the most part, the descriptions that are present in the *System V Programmer's Manual – STREAMS* provide the most definitive ipso facto standard for *STREAMS* implementation. In addition to this, some *STREAMS* implementations have provided some enhancements or restrictions over the *SVR 4* descriptions. Perhaps the most extensive embellishments have been provided for the *Solaris* implementation of *STREAMS*.

[1] One of the major deficiencies of the deprecated *LiS* implemntation was its inability to conform to the standards imposed on the user interface and behaviour of the Stream head.

Linux Fast-STREAMS has been implemented to provide maximum compatibility over a wide range of *STREAMS* implementations based on *SVR 4* and provides additional capabilities similar to those specific embellishments found in implementations such as *Solaris* through an add-on *STREAMS Compatibility* package.

The most delicate areas of compatibility across *STREAMS* implementations regard, not the use of *STREAMS* or DDI/DKI functions from within the *STREAMS* environment, but the invocation of *STREAMS* functions from outside the *STREAMS* environment. In particular, use of private locks and synchronization in the face of interrupts and external asynchronous callbacks is where implementations deviate the greatest. *Linux Fast-STREAMS* attempts to address these differences by providing a greater level of assurance and wider range of calling contexts for each of the *STREAMS* facilities.

Kernel interface compliance of the *Linux Fast-STREAMS* to *SVR 4* specifications is tested with custom validation test suites, test modules and test drivers that ship with the package. See Section 3.5.5 [Conformance Test Programs], page 35 for more information on conformance and validation test suites.

6.2 STREAMS Compatibility

Linux Fast-STREAMS provides a high degree of compatibility with other *STREAMS* implementation as listed below. Through the separate add-on *STREAMS Compatibility* package, source level compatibility is also provided.

— *SVR 3.2*
: *Linux Fast-STREAMS* provides a degree of operational compatibility with *SVR 3.2* to ease portability and common comprehension. Specific kernel utilities are provided by the *STREAMS Compatibility* package to provide full source level compatibility with *SVR 3.2*.

— *SVR 4.2 ES/MP*
: *Linux Fast-STREAMS* provides a high degree of operational compatibility with *SVR 4.2 ES/MP* to ease portability and common comprehension. Specific kernel utilities are provided by the *STREAMS Compatibility* package to provide full source level compatibility with *SVR 4.2 ES/MP*.

— *Mentat Portable STREAMS*
: *Linux Fast-STREAMS* provides a high degree of operational compatibility with *Mentat Portable STREAMS* to ease portability and common comprehension. Specific kernel utilities are provided by the *STREAMS Compatibility* package to provide full source level compatibility with *Mentat Portable STREAMS*.

— *AIX 5L Version 5.1*
: *Linux Fast-STREAMS* provides a high degree of operational compatibility with *AIX 5L Version 5.1* to ease portability and common comprehension. Specific kernel utilities are provided by the *STREAMS Compatibility* package to provide full source level compatibility with *AIX 5L Version 5.1*.

— *HP-UX 11.0i v2*
: *Linux Fast-STREAMS* provides a high degree of operational compatibility with *HP-UX 11.0i v2* to ease portability and common comprehension. Specific kernel

utilities are provided by the *STREAMS Compatibility* package to provide full source level compatibility with *HP-UX 11.0i v2*.

— *OSF/1 1.2/Digital UNIX/True 64*
Linux Fast-STREAMS provides a high degree of operational compatibility with *OSF/1 1.2/Digital UNIX* to ease portability and common comprehension.

— *UnixWare 7.1.3 (OpenUnix 8)*
Linux Fast-STREAMS provides a high degree of operational compatibility with *UnixWare 7.1.3 (OpenUnix 8)* to ease portability and common comprehension. Specific kernel utilities are provided by the *STREAMS Compatibility* package to provide full source level compatibility with *UnixWare 7.1.3 (OpenUnix 8)*.

— *Solaris 9/SunOS 5.9*
Linux Fast-STREAMS provides a high degree of operational compatibility with *Solaris 9/SunOS 5.9* to ease portability and common comprehension. Specific kernel utilities are provided by the *STREAMS Compatibility* package to provide full source level compatibility with *Solaris 9/SunOS 5.9*.

— *IRIX 6.5.17*
Linux Fast-STREAMS provides a high degree of operational compatibility with *IRIX 6.5.17* to ease portability and common comprehension. Specific kernel utilities are provided by the *STREAMS Compatibility* package to provide full source level compatibility with *IRIX 6.5.17*.

— *Mac OS 9 Open Transport*
Linux Fast-STREAMS provides a high degree of operational compatibility with *Mac OS 9 Open Transport* to ease portability and common comprehension. Specific kernel utilities are provided by the *STREAMS Compatibility* package to provide full source level compatibility with *Mac OS 9 Open Transport*.

— *SUPER-UX*
Linux Fast-STREAMS provides a high degree of operational compatibility with *SUPER-UX* to ease portability and common comprehension.

— *UXP/V*
Linux Fast-STREAMS provides a high degree of operational compatibility with *UXP/V* to ease portability and common comprehension.

— *LiS-2.16.18 and LiS 2.18.0*
Linux Fast-STREAMS provides a high degree of operational compatibility with *LiS 2.16.18* and *LiS 2.18.0* to ease portability and common comprehension. to ease portability and common comprehension. Specific kernel utilities are provided by the *STREAMS Compatibility* package to provide full source level compatibility with *LiS 2.16.18* and *LiS 2.18.0*.

For additional details, see Section "About This Manual" in *STREAMS Programmer's Guide*.

7 Releases

This is the OpenSS7 Release of the Linux Fast-STREAMS core, tools, drivers and modules that implement the *Linux Fast-STREAMS* SVR 4.2 MP STREAMS utility for Linux. This package is intended as a replacement package for *Linux STREAMS (LiS)*.

The following sections provide information on Linux Fast-STREAMS releases as well as compatibility information of OpenSS7 release to mainstream UNIX releases of the core, modules and drivers, as well as Linux kernel compatibility.

7.1 Prerequisites

The quickest and easiest way to ensure that all prerequisites are met is to download and install this package from within the *OpenSS7 Master Package*, `openss7-0.9.2.G`, instead of separately.

Prerequisites for the Linux Fast-STREAMS package are as follows:

1. *Linux* distribution, somewhat *Linux Standards Base* compliant, with a 2.4 or 2.6 kernel and the appropriate tool chain for compiling out-of-tree kernel modules. Most recent *Linux* distributions are usable out of the box, but some development packages must be installed. For more information, see Section 7.2 [Compatibility], page 61.
   - A fairly LSB compliant GNU/Linux distribution.[1]
   - Linux 2.4 kernel (2.4.10 - 2.4.27), or
   - Linux 2.6 kernel (2.6.3 - 2.6.26);
   - glibc2 or better.
   - GNU groff (for man pages).[2]
   - GNU texinfo (for info files).
   - GNU bison and flex (for config programs).
   - net-snmp (for SNMP agents).[3]

If you need to rebuild the package from sources with modifications, you will need a larger GNU tool chain as described in See Section 8.2.8 [Downloading from CVS], page 94.

7.2 Compatibility

This section discusses compatibility with major prerequisites.

7.2.1 GNU/Linux Distributions

Linux Fast-STREAMS is compatible with the following *Linux* distributions:[4]

- CentOS Enterprise Linux 3.4 (centos34) TBD
- CentOS Enterprise Linux 4.0 (centos4) TBD
- CentOS Enterprise Linux 4.92 (centos49) TBD

[1] See Section 7.2.1 [GNU/Linux Distributions], page 61, for more information.

[2] If you are using a Debian release, please make sure to install the groff extension package (‘`groff_ext`’), as it contains the `refer` or `grefer` commands necessary for including references in the manual pages.

[3] A wide range of net-snmp releases are supported, from UCD-SNMP 4.2.5 through net-snmp 5.4.

[4] Items marked as ‘`TBD`’ are scheduled to have support deprecated. That is, in a future release, the distributions marked ‘`TBD`’ will not longer be validated before release.

- CentOS Enterprise Linux 5.0 (centos5)
- CentOS Enterprise Linux 5.1 (centos51)
- CentOS Enterprise Linux 5.2 (centos52)
- Debian 3.0r2 Woody (deb3.0) TBD
- Debian 3.1r0a Sarge (deb3.1) TBD
- Debian 4.0r1 Etch (deb4.0)
- Debian 4.0r2 Etch (deb4.0)
- Debian 4.0r3 Etch (deb4.0)
- Fedora Core 1 (FC1) TBD
- Fedora Core 2 (FC2) TBD
- Fedora Core 3 (FC3) TBD
- Fedora Core 4 (FC4) TBD
- Fedora Core 5 (FC5) TBD
- Fedora Core 6 (FC6) TBD
- Fedora 7 (FC7)
- Fedora 8 (FC8)
- Fedora 9 (FC9)
- Gentoo 2006.1 (untested) TBD
- Gentoo 2007.1 (untested) TBD
- Lineox 4.026 (LEL4) TBD
- Lineox 4.053 (LEL4) TBD
- Mandrakelinux 9.2 (MDK92) TBD
- Mandrakelinux 10.0 (MDK100) TBD
- Mandrakelinux 10.1 (MDK101) TBD
- Mandriva Linux LE2005 (MDK102) TBD
- Mandriva Linux LE2006 (MDK103) TBD
- Mandriva One (untested)
- RedHat Linux 7.2 (RH7)
- RedHat Linux 7.3 (RH7)
- RedHat Linux 8.0 (RH8) TBD
- RedHat Linux 9 (RH9) TBD
- RedHat Enterprise Linux 3.0 (EL3) TBD
- RedHat Enterprise Linux 4 (EL4)
- RedHat Enterprise Linux 5 (EL5)
- SuSE 8.0 Professional (SuSE8.0) TBD
- SuSE 9.1 Personal (SuSE9.1) TBD
- SuSE 9.2 Professional (SuSE9.2) TBD
- SuSE OpenSuSE (SuSEOSS) TBD
- SuSE 10.0 (SuSE10.0) TBD

- SuSE 10.1 (SuSE10.1) TBD
- SuSE 10.2 (SuSE10.2) TBD
- SuSE 10.3 (SuSE10.3) TBD
- SuSE 11.0 (SuSE11.0)
- SLES 9 (SLES9) TBD
- SLES 9 SP2 (SLES9) TBD
- SLES 9 SP3 (SLES9) TBD
- SLES 10 (SLES10)
- Ubuntu 5.10 (ubu5.10) TBD
- Ubuntu 6.03 LTS (ubu6.03) TBD
- Ubuntu 6.10 (ubu6.10) TBD
- Ubuntu 7.04 (ubu7.04) TBD
- Ubuntu 7.10 (ubu7.10)
- Ubuntu 8.04 (ubu8.04)
- WhiteBox Enterprise Linux 3.0 (WBEL3) TBD
- WhiteBox Enterprise Linux 4 (WBEL4) TBD

When installing from the tarball (see Section 8.5.3 [Installing the Tar Ball], page 119), this distribution is probably compatible with a much broader array of distributions than those listed above. These are the distributions against which the current maintainer creates and tests builds.

7.2.2 Kernel

The *Linux Fast-STREAMS* package compiles as a *Linux* kernel module. It is not necessary to patch the *Linux* kernel to build or use the package.[5] Nor do you have to recompile your kernel to build or use the package. OpenSS7 packages use `autoconf` scripts to adapt the package source to your existing kernel. The package builds and runs nicely against production kernels from the distributions listed above. Rather than relying on kernel versions, the `autoconf` scripts interrogate the kernel for specific features and variants to better adapt to distribution production kernels that have had patches applied over the official kernel.org sources.

The *Linux Fast-STREAMS* package is compatible with 2.4 kernel series after 2.4.10 and has been tested up to and including 2.4.27. It has been tested from 2.6.3 up to and including 2.6.26 (with Fedora 9, openSUSE 11.0 and Ubuntu 8.04 patchsets). Please note that your mileage may vary if you use a kernel more recent than 2.6.26.4: it is difficult to anticipate changes that kernel developers will make in the future. Many kernels in the 2.6 series now vary widely by release version and if you encounter problems, try a kernel within the supported series.

UP validation testing for kernels is performed on all supported architectures. SMP validation testing was initially performed on UP machines, as well as on an Intel 3.0GHz Pentium IV 630 with HyperThreading enabled (2x). Because HyperThreading is not as independent

[5] At a later date, it is possible to move this package into the kernel, however, with continued resistance to STREAMS from within the *Linux* developer community, this is currently unlikely.

as multiple CPUs, SMP validation testing was limited. Current releases have been tested on dual 1.8GHz Xeon HP servers (2x) as well as dual quad-core SunFire (8x) servers.

It should be noted that, while the packages will configure, build and install against XEN kernels, that problems running validation test suites against XEN kernels has been reported. *XEN kernels are explicitly not supported.* This may change at some point in the future if someone really requires running OpenSS7 under a XEN kernel.

7.2.3 Architectures

The *Linux Fast-STREAMS* package compiles and installs on a wide range of architectures. Although it is believed that the package will work on all architectures supported by the Linux kernel being used, validation testing has only been performed with the following architectures:

- ix86
- x86_64
- ppc (MPC 860)
- ppc64

32-bit compatibility validation testing is performed on all 64-bit architectures supporting 32-bit compatibility. If you would like to validate an OpenSS7 package on a specific machine architecture, you are welcome to sponsor the project with a test machine.

7.2.4 Linux STREAMS

Linux Fast-STREAMS provides a suitable replacement for the (now deprecated) *Linux STREAMS (LiS) 2.18.0* package formerly maintained by Dave Goethe of GCOM.

7.3 Release Notes

The sections that follow provide information on OpenSS7 releases of the *Linux Fast-STREAMS* package.

Major changes for release streams-0.9.2.4

This is the thirteenth *OpenSS7 Project* release of *Linux Fast-STREAMS*. *LiS* was fully deprecated as of the previous release and *Linux Fast-STREAMS* is the only STREAMS package contained in the OpenSS7 Master Package (since '`openss7-0.9.2.D`').

This release is a stable, production grade release and is part of the OpenSS7 Master Package ('`openss7-0.9.2.G`'). The release includes maintenance support for recent distributions and tool chain, but also includes some performance and feature upgrades and inspection bug fixes. It deprecates previous releases. Please upgrade before reporting bugs on previous releases.

This release is primarily a maintenance release.

Major features since the last public release are as follows:

- Minor documentation corrections.
- Kernel module license made explicit "GPL v2". And then changed back to "GPL".
- License upgrade to *AGPL Version 3*.

- Modifications to build under *Fedora* '`2.6.22.5-49`' kernel. These changes also support '`2.6.22.9-91.fc7`' kernel.
- Added SNMP MIBs and AgentX sub-agent for STREAMS. This is a rather significant addition to Linux Fast-STREAMS which permits remote management of the entire STREAMS subsystem using SNMP. The agent is compatable with '`net-snmp`' and '`ucd-snmp`' back to '`ucd-snmp 4.2.5`'. This includes updates to the build process to support '`net-snmp`' packages all the way back to *RedHat 7.2*.
- Added the ability for the `sc(4)` STREAMS Configuration module to allow tuning STREAMS module info parameters and collect general purpose STREAMS statistics. Also, the module generates signals ('`SIGPOLL`') when STREAMS configuration changes. See `sc(4)` for more information.

  Also added a STREAMS configuration change notification registration function for use by `sc(4)`. See `streams_notify(9)`, `streams_register_notifier(9)` and `streams_unregister_notifier(9)` for more information.
- Ability to strap out major documentation build and installation primarily for embedded targets.
- Improvements to common build process for embedded and cross-compile targets.
- Support for '`flex 2.5.33`' in maintainer mode.
- Higher performance and updated performance paper.
- Fixed two POSIX compilance quirks and one soft-lockup when using `M_READ`. See 'BUGS' in the release for more information.
- Added workaround for use of `crash(8)`.
- Modifications to build under *Fedora* '`2.6.25-45.fc9`' and '`2.6.26.5-45.fc9`' kernels.
- Updated tool chain to '`m4-1.4.12`', '`autoconf-2.63`' and '`texinfo-4.13`'.
- Conversion of RPM spec files to common approach for major subpackages.
- Updated references database for manual pages and roff documents.
- Added test case 3.1.11.4 and modified test cases 3.2.1, 3.5.1 and 3.6.1 for the test-streams tests suite executable to test bug fixes and avoid regressions. See bug #016 and #017 in 'BUGS'.
- Build system now builds `yum(8)` repositories for RPMs and `apt-get(8)` repositories for DEBs. Installation documentation has been updated to include details of repository install sourcesref.
- Added `MODULE_VERSION` to all modules and drivers.

This is a public stable production grade release of the package: it deprecates previous releases. Please upgrade to the current release before reporting bugs.

As with other OpenSS7 releases, this release configures, compiles, installs and builds RPMs and DEBs for a wide range of Linux 2.4 and 2.6 RPM- and DPKG-based distributions, and can be used on production kernels without patching or recompiling the kernel.

This package is publicly released under the *GNU Affero General Public License Version 3*. The release is available as an `autoconf` tarball, SRPM, DSC, and set of binary RPMs and DEBs. See the downloads page for the `autoconf` tarballs, SRPMs and DSCs. For tarballs, SRPMs, DSCs and binary RPMs and DEBs, see the streams package page.

See http://www.openss7.org/codefiles/streams-0.9.2.4/ChangeLog and http://www.openss7.org/codefiles/streams-0.9.2.4/NEWS in the release for more information. Also, see the 'STREAMS.pdf' manual in the release (also in html http://www.openss7.org/STREAMS_manual.html).

For the news release, see http://www.openss7.org/rel20081029_K.html.

Major changes for release streams-0.9.2.3

This is the twelveth *OpenSS7 Project* release of *Linux Fast-STREAMS*. *LiS* was fully deprecated as of the previous release and *Linux Fast-STREAMS* is the only STREAMS package contained in the OpenSS7 Master Package (since 'openss7-0.9.2.D').

This release is a stable, production grade release. The release includes maintenance support for recent distributions and tool chain, but also includes some performance and feature upgrades and inspection bug fixes.

Major features since the last public release are as follows:

- Support build on openSUSE 10.2.
- Support build on Fedora 7 and 2.6.21 kernel.
- The strace(8), strerr(8) utilities and log(4) driver have had some corrections. The STREAMS trace logger is now an excellent way for trace logging of fielded production drivers. A number of OpenSS7 drivers have already been converted to use this facility.
- Support build on CentOS 5.0 (RHEL5).
- Support build on Ubuntu 7.04.
- Significant rework of the Stream head and both enabling and backenabling utilities. Handling of enabling flags QWANTR and QWANTW were not being performed quite properly. Also, service procedures for the read side stream head queue was added to defer wakeups when possible. The result is very significant performance improvements (like it wasn't fast enough already).

  STREAMS-based pipes in this package now perform 2 to 5 times (yes %200 to %500) faster than the old legacy 4.1BSD/SVR4 Linux pipes currently in the kernel: perhaps we should shout the "Fast" too.

  The impact of these performance changes is that *Linux Fast-STREAMS* now runs faster and looser on SMP systems: if your drivers have race conditions they will likely be exacerbated by this version.
- Corrected a few bugs. See BUGS in the release for more information.
- Significant rework of STREAMS syncrhonization. OSF/1-Mentat style syncrhonization levels, Solaris style perimeters, and SVR 4.2 style load refusal are fully supported. Some idiosynchracies of AIX, HP-UX, MacOT and VxWorks is also supported.
- Updated to gettext 0.16.1.
- Changes to support build on 2.6.20-1.2307.fc5 and 2.6.20-1.2933.fc6 kernel.
- Supports build on Fedora Core 6.
- Support for recent distributions and tool chains.

Major changes for release streams-0.9.2.2

This is the eleventh *OpenSS7 Project* release of *Linux Fast-STREAMS*. *LiS* was fully deprecated as of the previous release and *Linux Fast-STREAMS* is now the only STREAMS package contained in the OpenSS7 Master Package (since '`openss7-0.9.2.D`').

This release is a stable, production grade release for *Linux Fast-STREAMS*. The release is primarily a maintenance release to support recent distributions and tool chain.

Major features since the last public release are as follows:

- Fix for clone open failure locking problems, demand loading of clone minors, an error in clone minor device deregistration, and an error in queue syncrhornization. See `http://www.openss7.org/codefiles/streams-0.9.2.4/BUGS` in the release for more information.
- Added feature to perform automatic reference counting of modules for the `esballoc(9)` free routines on Linux 2.6 kernels. See new `esballoc(9)`.
- Added versions to all exported symbols. Made LFS unique functions GPL export.
- Improvements to the common build environment with better support for standalone package builds on 2.4 kernels.
- Support for autoconf 2.61, automake 1.10 and gettext 0.16.
- Support for Ubuntu 6.10 distribution and bug fixes for i386 kernels.

Major changes for release streams-0.9.2.1

This is the tenth *OpenSS7 Project* release of *Linux Fast-STREAMS*. The release number has been moved from the 0.9a sequence to 0.9.2 to indicate that the package has moved to a production grade. *LiS* has been fully deprecated by this release and *Linux Fast-STREAMS* is now the only STREAMS package contained in the OpenSS7 Master Package ('`openss7-0.9.2.D`').

This release is a stable, production grade release for *Linux Fast-STREAMS*. The release is primarily a maintenance release. Some minor defect corrections have been applied, but no significant development has occurred. The release provides the following enhancements and fixes:

- Testing of, and a few bug corrections to, the strlog() feature. Trace and error logging working well.
- Support for most recent 2.6.18 kernels (including Fedora Core 5 with inode diet patch set).
- The package now builds a replacement libLiS and libpLiS library so that user applications written to work with LiS-2.18.1 through LiS-2.18.7 do no need to be recompiled. Added versions to all library symbols in all three libraries.
- Minor bug fixes to STREAMS library. isastream(3) and fattach(3) must not contain an asynchronous thread cancellation point, but they could. Added asynchronous thread cancellation protection to these functions to remove any thread cancellation points.
- Now builds 32-bit compatibility libraries, as well, and tests them against 64-bit kernel modules and drivers. The '`make installcheck`' target will now automatically test both 64-bit native and 32-bit compatibility versions, one after the other, on 64-bit platforms.

- Many documentation updates for all OpenSS7 packages. Automated release file generation making for vastly improved and timely text documentation present in the release directory.
- Dropped support for *LiS*.
- Updated `init` scripts for proper addition and removal of modules.
- Start assigning majors at major device number 231 instead of major device number 230. Assign major device number 230 explicitly to the clone device. Package will now support extended ranges of minor devices on 2.6 kernels under *Linux Fast-STREAMS* only. *streams* now supports expanded addressable minor device numbers, permitting 2^16 addressable minor devices per major device number on 2.6 kernels: *LiS* cannot support this change.
- Better detection of SUSE distributions, release numbers and SLES distributions: support for additional *SuSE* distributions on `ix86` as well as `x86_64`. Added distribution support includes *SLES 9*, *SLES 9 SP2*, *SLES 9 SP3*, *SLES 10*, *SuSE 10.1*.
- Improved compiler flag generation and optimizations for recent `gcc` compilers and some idiosyncratic behaviour for some distributions (primarily SUSE).
- Optimized compilation is now available also for user level programs in addition to kernel programs. Added new '`--with-optimize`' option to `configure` to accomplish this.
- Added `--disable-devel configure` option to suppress building and installing development environment. This feature is for embedded or pure runtime targets that do not need the development environment (static libraries, manual pages, documentation).
- Added `send-pr` script for automatic problem report generation.
- The package will now build doxygen(1) html documentation with the 'doxy' make target. See 'make help' or README-make in the distribution for more information.

Major changes for release streams-0.7a.6.rc3

Third release candidate.

- Minor bug fixes to STREAMS library. isastream(3) and fattach(3) must not contain an asynchronous thread cancellation point, but they could. Added asynchronous thread cancellation protection to these functions to remove any thread cancellation points.
- The package will now build doxygen(1) html documentation with the 'doxy' make target. See 'make help' or README-make in the distribution for more information.
- Now builds 32-bit compatibility libraries and tests them against 64-bit kernel modules and drivers. The '`make installcheck`' target will now automatically test both 64-bit native and 32-bit compatibility versions, one after the other, on 64-bit platforms.
- Added versions to all library symbols.
- Automated release file generation making for vastly improved and timely text documentation present in the release directory.
- Many documentation updates for all OpenSS7 packages.
- This release candidate includes the changes made to the strsctp drivers at the *2006 SCTP Interop* at the *University of British Columbia*. This version was interoperability tested with all implementations present.

- This release candidate provides support for additional *SuSE* distributions on `ix86` as well as `x86_64`. Added distribution support includes *SLES 9*, *SLES 9 SP2*, *SLES 9 SP3*, *SLES 10*, *SuSE 10.1*.
- Better detection of SUSE distributions, release numbers and SLES distributions.
- Optimized compilation is now available also for user level programs in addition to kernel programs. Added new '`--with-optimize`' option to `configure` to accomplish this.
- Improved compiler flag generation and optimizations for recent `gcc` compilers and some idiosyncratic behaviour for some distributions (primarily SUSE):
  - Remove '`-fno-reorder-blocks`' and '`-fno-reorder-functions`' options added by some recent 2.6 '`Makefile`'s for '`x86_64`' architecture: it impedes performance optimizations.
  - Remove '`-ffunction-sections`' option added by some recent 2.6 '`Makefile`'s for '`x86_64`' architecture: this is an insane option and should never have been used.
  - Add '`-ffreestanding`' that some older 2.6 '`Makefile`'s (such as that with *SLES 9 2.6.5* kernel) neglect to add to the `gcc` command line.
  - *SLES 10* expands the directory before '`autoconf.h`' on the `gcc` command line for some reason. `configure` script watches out for this now.
- Updated `init` scripts for proper addition and removal of modules.
- Start assigning majors at major device number 231 instead of major device number 230. Assign major device number 230 explicitly to the clone device.
- *streams* now supports expanded addressable minor device numbers, permitting 2^16 addressable minor devices per major device number on 2.6 kernels: *LiS* cannot support this change.

This release is an internal release candidate and was not publicly released.

Major changes for release streams-0.7a.6.rc2

Second release candidate.

This release candidate also contains the results of performance testing of the new second generation *UDP* driver (implemented completely in *STREAMS* instead of using an internal socket).

This release candidate also contains support for *SuSE 10.1*.

This release is an internal release candidate and was not publicly released.

Major changes for release streams-0.7a.6rc1

Release candidate for Mark Fugate.

Added '`--enable-devel`' `configure` option for embedded targets. Added `send-pr` script for automatic problem report generation.

This release is an internal release candidate and was not publicly released.

Major changes for release streams-0.7a.5

This release is primarily to support additional compilers (gcc 4.0.2), architectures (x86_64, SMP, 32-bit compatibility), recent Linux distributions (EL4, SuSE 10, LE2006, OpenSuSE) and kernels (2.6.15).

- Changes to wait queues. Split single wait queue into four independent wait queues. Reworked wait queues for both old style (2.4) and new style (2.6) semantics.
- Changes to satisfy gcc 4.0.2 compiler.
- Corrected build flags for Gentoo and 2.6.15 kernels as reported on mailing list. Build and run tested on FC4 i686 and x86_64 kernels based on 2.6.15.
- Corrections for and testing of 64-bit clean compile and test runs on x86_64 architecture. Some bug corrections resulting from gcc 4.0.2 compiler warnings.
- Initial corrections for and testing of SMP operation on Intel 630 Hyper-Threaded SMP on x86_64. This package should now run well on N-way Xeons even with Hyper-Threading enabled.
- Corrections and validation of 32-bit compatibility over 64-bit on x86_64. Should apply well to other 64-bit architectures as well.

This is a public beta test release of the package.

Major changes for release streams-0.7a.4

This is primarily a bug fixes release and corrections resulting from testing.

Major changes for release streams-0.7a.3

With this release version numbers were changed to reflect an upstream version only to be consistent with other OpenSS7 package releases. All *RPM* release numbers will be '`-1$(PACKAGE_RPMEXTRA)`' and all *Debian* release numbers will be '`_0`'. If you wish to apply patches and release the package, please bump up the release number and apply a suitable release suffix for your organization. We leave *Debian* release number '`_1`' reserved for your use, so you can still bundle the source in the '`.dsc`' file.

Major changes for this release include build against Linux 2.6 kernels and popular distributions based on the 2.6 kernel as well as wider distribution support.

This was an internal beta test release and was not released publicly.

Major changes for release streams-0.7a-3

Updates to common build process. Documentation updates.

This was an internal alpha test release and was not released publicly.

Major changes for release streams-0.7a-2

Removed all XTI/TLI and Linux networking code, headers and documentation from streams distribution and set epoch at 0. Linux networking code has been migrated to the strxnet, strinet and strsctp packages. The purpose for doing this was to allow the Linux networking to build against *Linux Fast-STREAMS* as well as *Linux STREAMS* and is a preparation for phasing out LiS and phasing in LfS.

This was an internal alpha test release and was not released publicly.

Initial release streams-0.7a-1

This is the initial release of the Linux Fast-STREAMS package for Linux. This is intended as a high-performance, production replacement for *Linux STREAMS (LiS)*. Linux Fast-STREAMS has the following features:

- optimized for Linux kernels.
- prepared for mainstream Linux kernel adoption.
- lindented and follows kernel coding practises.
- compatibility modes for AIX, HPUX, OSF, Solaris, UnixWare, SVR 4.2 and LiS.
- supports all major SVR4.2 variants.
- licensed under GPL with commercial licensing available.
- supports full SVR 4.2 MP synchronization models.
- runs at SoftIRQ.
- provides common SVR 4.2 system tunable parameters and sysctls.
- provides /proc file system access for debugging and performance tuning.
- provides a full set of common STREAMS modules and drivers.
- provides full name-streams device and shadow special file system support.

This was an internal alpha test release and was not released publicly.

7.4 Maturity

The *OpenSS7 Project* adheres to the following release philosophy:

- pre-alpha release
- alpha release
- beta release
- gamma release
- production release
- unstable release

7.4.1 Pre-Alpha Releases

Pre-alpha releases are releases that have received no testing whatsoever. Code in the release is not even known to configure or compile. The purpose of a pre-alpha release is to make code and documentation available for inspection only, and to solicit comments on the design approach or other characteristics of the software package.

Pre-alpha release packages ship containing warnings recommending that the user not even execute the contained code.

7.4.2 Alpha Releases

Alpha releases are releases that have received little to no testing, or that have been tested and contains known bugs or defects that make the package unsuitable even for testing. The purpose for an *alpha* release are the same as for the pre-alpha release, with the additional purpose that it is an early release of partially functional code that has problems that an external developer might be willing to fix themselves and contribute back to the project.

Alpha release packages ship containing warnings that executing the code can crash machines and might possibly do damage to systems upon which it is executed.

7.4.3 Beta Releases

Beta releases are releases that have received some testing, but the testing to date is not exhaustive. *Beta* release packages do not ship with known defects. All known defects are resolved before distribution; however, as exhaustive testing has not been performed, unknown defects may exist. The purpose for a *beta* release is to provide a baseline for other organizations to participate in the rigorous testing of the package.

Beta release packages ship containing warnings that the package has not been exhaustively tested and that the package may cause systems to crash. Suitability of software in this category for production use is not advised by the project; however, as always, is at the discretion of the user of the software.

7.4.4 Gamma Releases

Gamma releases are releases that have received exhaustive testing within the project, but external testing has been minimal. *Gamma* release packages do not ship with known defects. As exhaustive internal testing has been performed, unknown defects should be few. Please remember that there is NO WARRANTY on public release packages.

Gamma release packages typically resolve problems in previous *beta* releases, and might not have had full regression testing performed. Suitability of software in this category for production use is at the discretion of the user of the software. *The OpenSS7 Project* recommends that the complete validation test suites provided with the package be performed and pass on target systems before considering production use.

7.4.5 Production Releases

Production releases are releases that have received exhaustive testing within the project and validated on specific distributions and architectures. *Production* release packages do not ship with known defects. Please remember that there is NO WARRANTY on public release packages.

Production packages ship containing a list of validated distributions and architectures. Full regression testing of any maintenance changes is performed. Suitability of software in this category for production use on the specified target distributions and architectures is at the discretion of the user. It should not be necessary to preform validation tests on the set of supported target systems before considering production use.

7.4.6 Unstable Releases

Unstable releases are releases that have received extensive testing within the project and validated on a a wide range of distributions and architectures; however, is has tested unstable and found to be suffering from critical problems and issues that cannot be resolved. Maintenance of the package has proved impossible. *Unstable* release packages ship with known defects (and loud warnings). Suitability of software in this category for production use is at the discretion of the user of the software. *The OpenSS7 Project* recommends that the problems and issues be closely examined before this software is used even in a non-production environment. Each failing test scenario should be completely avoided by the application. *OpenSS7* beta software is more stable that software in this category.

7.5 Bugs

7.5.1 Defect Notices

Linux Fast-STREAMS could possibly contain unknown defects. This is a *production* release. Nevertheless, some remaining unknown defects could possibly be harmful. Validation testing has been performed by the *OpenSS7 Project* and external entities on this software for the set of systems listed in the release notes. Nevertheless, the software might still fail to configure or compile on other systems. The *OpenSS7 Project* recommends that you **validate this software for your target system before using this software**. Use at your own risk. Remember that there is **NO WARRANTY**.[6]

This software is *production* software. As such, it is stable on validated systems but might still crash your kernel in unique circumstances. Installation of the software on a non-validated distribution might mangle your header files or Linux distribution in such a way as to make it unusable. Crashes could possibly lock your system and rebooting the system might not repair the problem. You can possibly lose all the data on your system. Because this software stands a chance of crashing your kernel, the resulting unstable system could possibly destroy computer hardware or peripherals making them unusable. You might void the warranty on any system on which you run this software. YOU HAVE BEEN WARNED.

7.5.2 Known Defects

With the exception of packages not originally created by the *OpenSS7 Project*, the *OpenSS7 Project* software does not ship with known bugs in any release stage except *pre-alpha*. *Linux Fast-STREAMS* had no known bugs at the time of release.

7.5.3 Defect History

This section contains historical bugs that were encountered during development and their resolutions. This list serves two purposes:

1. It captures bugs encountered between releases during development that could possibly reoccur (and the Moon is made of blue cheese). It therefore provides a place for users to look if they encounter a problem.
2. It provides a low overhead bug list between releases for developers to use as a '`TODO`' list.

Bugs

`025. 2008-10-17T05:57:29+0000`

'`putnext(q, mp)`' was checking whether procedures had been turned off on queue '`q`'. This was not correct as it is only the '`q->q_next`' put procedure that would be executed. It should only check procedures on '`q->q_next`'.

fixed in `streams-0.9.2.4`

`024. 2008-10-11T19:36:41+0000`

A list delete corruption bug in the STREAMS driver and module lookup functions (e.g. `__cdrv_lookup`) was discovered by the list debugging in the *FC9* kernel.

fixed in `streams-0.9.2.4`

[6] See sections **Disclaimer of Warranty** and **Limitation of Liability** under [GNU Affero General Public License], page 144.

023. 2008-10-11T19:36:23+0000
: Not really a bug, but newer (2.6.25) kernels no longer permit registration of binary identifiers for sysctls (i.e. *ctl_name*). The proc filesystem entries (i.e. *procname*) are still permitted and *ctl_name* should be set to zero for these kernels. Added a check for the existence of symbol `sysctl_check_table()` to identify when binary registration is forbidden. Another related problem is that when binary registration of system controls is not possible, `sysctl(2)` becomes worthless. Unfortunately, the STREAMS MIB agent was written to use `sysctl(2)` and needs to be rewritten to use the '`/proc/sys`' filesystem instead ala `sysctl(8)`.

  fixed in `streams-0.9.2.4`

022. 2008-10-07T18:40:25+0000
: When overriding 32-bit compatability on input-output controls conflicting from the CDROM block device with STREAMS input-output controls, the override was not properly passing CDROM input-output controls through due to a missing break statement in the override loop. This bug affected pre-2.6.11 kernels, likely manifesting itself in a non-function CDROM device while STREAMS was loaded. Bug reported and one-line fix provided by Sylvain Chouleur for *DGAC*.

  fixed in `streams-0.9.2.4`

021. 2008-08-01T22:32:08+0000
: When flushing queues the backenable bits were not being initialized to zero in `__flushq()`, resulting in back-enabling of bands (or the normal queue) was being performed depending on the uninitialized values in the backenable bit array. This only affected `I_SETSIG` signals for `SWRNORM` and `SWBAND`, and the only when flushing queues. Fix properly initializes the backenable array.

  fixed in `streams-0.9.2.4`

020. 2008-07-31T04:59:41+0000
: Not really a bug (for *STREAMS*), but when the '`streams.ko`' kernel module is loaded, the `crash(8)` debugger will not debug a running kernel because it finds the `runqueues()` exported function in the '`streams.ko`' module instead of the the static one from the kernel. This has been temporarily renamed by macro to `srunqueues()` (notice the leading '`s`') until `crash(8)` learns to do the right thing and check that the symbol it looks up comes from the kernel instead of a kernel module.

  workaround in `streams-0.9.2.4`

019. 2008-07-25T22:41:47+0000
: When `M_READ` was being issued by the Stream Head downstream an `srlock()` imbalance in `strsendmread()` was causing soft-lockups on close for recent read-write lock implementations on *CentOS 5.2* for '`x86_64`'.

  fixed in `streams-0.9.2.4`

018. 2008-07-25T01:15:26+0000
: Previous fix didn't work too good: returning [`EAGAIN`] when hung-up on `getmsg(2)`, `getpmsg(2)`, `read(2)`, `readv(2)` instead of 0 and terminal end of file. This caused a regression on four or five other test cases.

fixed in `streams-0.9.2.4`

`017. 2008-04-10T15:17:30+0000`

When `M_DATA` is sent upstream followed by `M_HANGUP`, `read(2s)` is returning zero (0) and not permitting the data associated with the `M_DATA` to be read. This is a bug per documentation. `read(2s)` should operate as normal following a hangup until all data is read and then return zero (0).

The difficulty is that when waking up from a read sleep or when entering read the hangup condition was generating an internal [`ESTRPIPE`] error. This was altered so that [`ESTRPIPE`] is only returned during the hangup condition after the read queue has been tested and the caller is about to sleep on read.

Test cases 3.2.1, 3.5.1 and 3.6.1 in the test-streams test suite executable were altered to validate the fix for this case and curtail regressions.

fixed in `streams-0.9.2.4`

`016. 2007-11-14T17:23:57+0000`

Read is blocking when data has been read, `O_NONBLOCK` and `O_NDELAY` unset, `RFILL` unset, in non-SVR4 mode. This violates POSIX specifications.

Test case 3.1.11.4 in the test-streams test suite executable was generated to validate the fix for this case and to curtail regressions.

fixed in `streams-0.9.2.4`

`015. 2007-11-14T17:19:01+0000`

Dynamic allocation of major device numbers is not working on recent 2.6 kernels. Someone slipped some code in the kernel to have register_chrdev() allocate from major 255 down (again). Changed code to allocate modid according to our own rules and then request the same for a major device number. This also ensures that module ID and major are the same.

fixed in `streams-0.9.2.4`

`014. 2007-05-17T21:48:24+0000`

The `dupb(9)` utility had an obnoxious bug where it permitted the *db_ref* count to wrap to zero, causing buffer allocation and freeing problems. This was very difficult to debug. `dupb(9)` now fails if the reference count has reached 255. When `dupb(9)` fails, the user should check if the reference count has reached 255, and if it has, attempt a deep `copyb(9)` instead. At some point it might be useful to have STREAMS do the deep copy automatically. This was discovered in '`strsctp`' loopback tests where message blocks are rapidly duplicated for retransmission.

fixed in `streams-0.9.2.3`

`013. 2007-05-17T21:48:06+0000`

The log driver, strace, strerr and strclean utilities had some bugs. The '`strsctp`' driver now makes extensive use of `strlog(9)` trace and error logging and the log driver and utilities have been corrected. These facilities are now production grade.

fixed in `streams-0.9.2.3`

012. 2007-04-13T01:47:30+0000

It appears that *Ubuntu 6.10* has a rather broken implementation of the LSB `install_init` that has been inherited from Debian (a python script, none the less). This implementation refuses to properly install a disabled service (one with an empty or missing Default-Start: tag), but, rather invokes updated-rc.d in such a way that the init script is started at runlevels '`2 3 4 5`' instead. This was causing problems with the strace and strerr services which are normally installed disabled.

This uncovered the fact that the Debian-style init scripts were not working anyway. The scripts have been fixed and the strace and strerr utilities now default to enabled.

fixed in `streams-0.9.2.3`

011. 2007-04-10T10:56:42+0000

The strbcflag flag was never being cleared, causing infinite looping of the scheduler once the maximum number of buffers was reached. This also revealed a problem that bufcalls were being run unncecessarily (when strbcwait was set, instead of only when strbcflag was set).

fixed in `streams-0.9.2.3`

010. 2007-04-10T10:55:29+0000

The stream event sequence number was wrapping and becoming larger than the event mask resulting in inability to cancel buffer callbacks and timeouts.

fixed in `streams-0.9.2.3`

009. 2007-04-02T11:57:35+0000

'`ldl`' was using an incorrect MKDEV command, but when the Stream head attempted to redirect the open to the new (mangled) major device number, it properly returned ENXIO, but did not release a reference to the module. Need to check code paths for this to see where the reference needed to be released.

known bug

008. 2007-03-31T05:33:29-0600

When loosening SMP locking, found a bug in the QWANTR handling in getq() and back-enabling in flushq() and flushband(). Both of these were generating false back-enables. The getq() was generating a *lot* of false back-enables. Whenever getq() found an empty queue it was not only setting QWANTR, but it was back-enabling the queue. The result is that if service procedures are used exclusively (that is, qi_put() always does a putq()), getq() would generate a false back-enable for each message. Also, the enabled queue would generate another false back-enable. Significant performance gains should be noticed.

fixed in `streams-0.9.2.3`

007. 2007-03-16T17:33:20-0600

Jérémy Compostella pointed out an error in strallocpmsg() where it was always assigining M_PCPROTO to messages created with I_FDINSERT.

fixed in `streams-0.9.2.3`

`006. 2007-03-14T23:48:26-0600`

There appears to be an inode lock imbalance that occurred for several clone error paths in stropen. If the returned major device number does not correspond to a driver, or an snode cannot be acquired for the new entry and the stream head reparented.

fixed in `streams-0.9.2.2`

`005. 2007-03-07T15:53:06-0700`

Demand loading of kernel modules for clone devices opened, for example, as /dev/streams/clone/mux was requesting module streams-clone-mux and /dev/streams/clone/mux but was not requesting streams-mux or /dev/streams/mux and the modules were failing to demand load.

fixed in `streams-0.9.2.2`

`004. 2007-02-26T08:25:09-0700`

Jérémy Compostella pointed out error in clone.c. When an automatic clone minor device was unregistered, it was unregistering the modid instead of the major number. This was not noticed because all OpenSS7 drivers have the same modid as major number (strconf does this automatically).

fixed in `streams-0.9.2.2`

`003. 2007-02-26T08:25:09-0700`

Jérémy Compostella pointed out syntax error in strsched.c that kept synqs from compiling properly.

fixed in `streams-0.9.2.2`

`002. 2006-09-24T20:02:00+0000`

Discovered asynchronous thread cancellation inconsistencies in libLiS libpLiS by inspection during documentation. `isastream(2)`, `fattach(2)` were not performing proper asynchronous thread cancellation suppression so that these function contained a cancellation point when the should not.

fixed in `streams-0.7a.6.rc3`

`001. 2006-07-05T21:54:49+0000`

Fedora Core 5 reports a rwlock bug during udp module unloading as follows:

```
BUG: rwlock wrong CPU on CPU#0, rmmod/7515
Call Trace:
  {rwlock_bug+100}
  {_raw_write_unlock+88}
  {:streams:unregister_strnod+211}
  {:streams:unregister_clone+64}
  {:streams:unregister_strdev+24}
  {:streams_udp:udpterminate+26}
  {sys_delete_module+406}
  {system_call+126}
```

It appears that unregister_strnod() is scheduling while holding a write lock on cdevsw_lock. This is probably in iput() called within cmin_del.

fixed in `streams-0.7a.6.rc2`

There were a number of places where sleeping functions were called with spin-locks held, causing the CPU awaking from the sleep to sometimes be different from the CPU that took the lock. This was buggy, so I reworked all of these cdev and fmod sections to handle spin locks properly. FC5/SMP on HT no longer reports these bugs.

7.6 Schedule

Current Plan

There are not many things left to be done on the production Linux Fast-STREAMS package. As of the streams-0.9.3 release, performance modifications are complete. The package now exhibits performance on STREAMS-based pipes and TPI drivers that is significantly (factor of 2 or more) superior to that experienced by legacy Linux facilities.

Therefore, the current plan for Linux Fast-STREAMS is largely a maintenance plan. Items on the todo list, below, will be picked up as time permits. The OpenSS7 Project intends to release regularly new versions of Linux Fast-STREAMS that build and validate against upcoming releases of the supported Linux Distributions available from major distributors and upcoming releases of the Linux kernel, both mainline and as patched by major distributors. This release schedule is approximately every 3 to 6 months. More recent corrections and support for new distributions and kernels can be obtained by sponsoring the OpenSS7 Project and obtaining access to the live CVS repository (also available as a git repository).

One development activity in the works for Linux Fast-STREAMS is to provide integral support for more embedded cross-platform development systems such as the Denx ELDK, as well a existing and emerging RT kernels such as Montavista and the upcoming SuSE and RedHat RT kernels. This is a significant undertaking and will only be embarked upon when the OpenSS7 Project is given free access to these RT kernels and distributions.

Things to Do

- Support for RT kernels. This is a little more than just having the STREAMS scheduler run as a non-RT process kernel thread, which it does now, and which is trivial. (The existing package should compile and run against these kernels with minor modification in this event.)

  More to the point is working the light-weight STREAMS scheduler and service procedures into a prioritized scheme where service procedures run as real-time, yet preemptable tasks. In contrast to the current scheme, it is likely that the approach would be to either spawn multiple kernel threads for the STREAMS scheduler at different priorities, or to alter the priority of the STREAMS scheduler in response to the scheduling of specific queues at specific priorities. A design is not really possible until the intricacies of upcoming RT kernels are discovered.

  TODO: Provide support for RT kernels.

- Per cpu data:- I am still using the older approach of using cache line aligned arrays for per-cpu data. This, of course, does not fully utilize NUMA architectures. For NUMA architectures we need to use the per-cpu utilities provided by the 2.6 kernel. I haven't touched converting this yet.

Also, there are several NUMA supporting STREAMS utility functions (allocb_node, etc.) that need to be supported yet.

TODO: Convert cacheline aligned arrays to NUMA per-cpu data on 2.6 kernels. Complete NUMA supporting STREAMS facilities.

– Split '`include/sys/streams/stropts.h`' by architecture. There is conflicting numbering on the standard *STREAMS* input-output controls:

```
I_SWROPT(7)       I_GWROPT(7)       I_LIST(7)
I_FLUSHBAND(7)    I_CKBAND(7)       I_GETBAND(7)
I_ATMARK(7)       I_SETCLTIME(7)    I_GETCLTIME(7)
I_CANPUT(7)
```

System V Release 4 UNIX® vendors use one set and *OSF* UNIX® vendors use another. Namely *HP-UX*, *OSF/1.2*, *AIX*, *Mac OpenTransport* use OSF numbering, whereas *IRIX*, *Solaris*, *UnixWare* and others use SVR4 numbering. So, for HPPA, Alpha, PowerPC, we should use the OSF numbering.

I know that it is a fall-back to the SVR4 way of separating architectural differences by UNIX vendor (if it is HPPA, it must be sold by HP and it must be HP-UX running on it, for example), but even the Linux kernel is victim to this (many ioctls and some errno numbering is split this way). It is completely entrenched in GNU autoconf's config.guess.

TODO: Split '`include/sys/streams/stropts.h`' by processor architecture.

– A similar numbering mismatch occurs for many of the message block types.

TODO: Split '`include/sys/streams/streams.h`' by processor architecture.

– Implement `I_EGETSIG(7)` and `I_ESETSIG(7)`. These are *Solaris* enhanced version of the `I_GETSIG(7)` and `I_SETSIG(7)` *STREAMS* input-output controls. The difficulty with their implementation is that the entire signal handling setup inside the Stream head code is geared toward the calling process and needs to be adjusted to be general enough for any process or process group. Until then, *Linux* file asynchronous I/O is supported.

PARTLY DONE: Wrote the manual pages and added them to the build. Placed function skeletons that return [`EOPNOTSUPP`] for these functions in the Stream head.

TODO: Implement `I_EGETSIG(7)` and `I_ESETSIG(7)`.

– It is possible on 2.6 kernels to use the ability to determine the module that owns a function to perform module reference counting for `esballoc(9)` callback functions. That is, when `esballoc(9)` (and friends) are called, the module owning the callback function has its module reference count incremented. When the block is freed and the callback function returns, the module has its module reference count decremented. The pertinent kernel function is module_text_address() that returns the module in which a text address resides.

DONE: Module reference counting performed when module_text_address() is available.

– Add a tail padding amount to the stream head as a option to facilitate conversion of mblks to sk_bufs.

DONE: Implemented as SO_WRPAD.

– Socket buffer handling:

1. Rather than write offset and padding, why not provide a flag (e.g. SO_SKBUFF) to indicate to the stream head to allocate an sk_buff with the message block and share buffers between mblk and sk_buff, then, the sk_buff can be used without allocation in the bottom half. esballoc() and alloc_skbuff() can be used to set up the message block. dup() could be made aware of the hidden sk_buff and increment the shared sk_buff count as well. Also, msgpullup() and pullupmsg() could be made aware of message blocks containing sk_buffs and have them do the appropriate thing.
2. The other thing that is needed is some way to tell the other end of a loopback connection that the sk_buff it has received already has an mblk attached to it as above. Then the message block could be simply passed upstream and one would not need to be esballoc'ed for it.
3. Another thing is to provide the ability to partial checksum and copy data from user into these sk_buffs, but setting an SO_CSUM flag along with the SO_SKBUFF flag to indicate the type of checksum to perform.

The combination of the above three items should provide some serious performance gains for Linux networking based stream heads.

PARTIALLY DONE:
Item (1) is done and complete. The 2nd generation UDP and RAW drivers are already using it. Item (2) and (3) remain.

– Had another look at specfs, devfs and udev. It looks like we can create minor device nodes within /dev (not just /dev/streams) using devfs or udev. Again, this doesn't do everything that specfs does. specfs will demand load when an attempt is made to open a non-existent character device. Nevertheless, we can describe a "streams" class for udev and when a module registers a minor device node, we can have udev create that device node and provide permissions by adding our files to the /etc/udev/rules.d and /etc/udev/permissions.d directories.

Therefore, on a udev system, we should make strconf-sh create the necessary rules.d and permissions.d file entries. register_strnod will be modified to create a udev instance within the stream class matching the rules.d and permissions.d entry when creating a minor device node within the specfs.

On a devfs system, register_strdev and register_strnod should perform devfs calls instead of calling register_chrdev. That way minor device nodes will automatically appear at least once the module is loaded.

TODO: rationalize specfs to devfs and udev

– Have the STREAMS subsystem register a panic notifier on 2.6 kernels to be able to recover from panics caused by misbehaving STREAMS modules or drivers.

TODO: Register panic notifier.

– Timers and Buffer callbacks:- Still haven't tested these.

DONE: Timers are working nicely for SCTP that was tested at the Vancouver interop. Had no problems whatsoever. Probably didn't run into a buffer callback, though, so those need to be more rigorously tested.

Even mi_timers on M2PA are working fine.

- More performance testing and profiling on SMP. On the same kernel running non-SMP we get pipe performance of about 80-90% of a Linux native pipe. Just running an SMP kernel drops this to 60% comparative. Running both CPUs in an SMP kernel does not improve matters. Need to profile this up 80-90% on SMP too. Also, so many changes were made for 64bit and 32bit compatibility that the old profiling information is out of date and needs to be updated.

  DONE: Profile and performance tests on SMP.

- I am interested to convert the perftest program to use a FIFO instead of a pipe. This is because a FIFO is more closely related to a Linux Native pipe (i.e it has a read side and a write side, is really only one file pointer, and only supports unidirectional flow). Comparative tests as opposed to STREAMS-based pipes should be interesting.

  DONE: Tested. Results were unimpressive. STREAMS FIFOs perform about 1% better than STREAMS-based pipes.

- 32bit compatibility:- Not done yet, but a plan in place. Override stupid CDROM ioctl conversions on kernels before 2.6.11, use compat_ioctl after that. For the older read/write interface it will be necessary to have two "magic" lengths: one the same as the old one for 32-bit and a new 64-bit "magic" length. This is so that the internal function can convert. Perhaps it can really be the same number in the lower 32-bits. Note that on later kernels there is a CONFIG_COMPAT define that we might want to check in the configure script.

  DONE: Tested on x86_64 with i686.

  DONE: Documentation of new registration functions.

- SMP:- Finally got at least a Hyper-Threaded Intel 630 for testing. There are some issues discovered when running the test suites. Initial debugging is done (everything runs and doesn't crash) it is just that some multiple writers and readers are getting stuck in wait queues. What we need to do is to split the wait queues into open, close, read, write and ioctl from the big wait queue that it is, and largely get rid of the RSLEEP, WSLEEP, IOCWAIT type bits (we could still set them for compatibility with SVR4 but not examine them).

  DONE: Tested on X86_64 SMP.

- 64bit clean:- Pretty good now. I have clean compiles and test suite runs on 2.6.9-22.EL x86_64 kernels.

  DONE: Tested on x86_64.

- LiS Binary Compatibility:- Pretty much messed up right now. I have the regparm(0) stuff on the callout and callback functions, and the STREAMS Compatibility Modules does regparm(0) on all the lis_ functions. Primary STREAMS data structures align on the first portion of the data structure. Sizes vary, so don't do q+1. Flags are a little different. There are however several problematic data structures: cred_t and the ioctypes: iocblk, copyreq, copyresp, and the linkblk. When you enable binary compatibility mode, it uses the LiS versions of these.

  DONE: Needs testing.

- LfS Binary Compatibility:- A little better. Callouts and callbacks always regparm(0). STREAMS utility functions are always regparm(3). Data structures are stable for the most part. cred_t, however, is variable depending on the kernel. Compile with the LiS binary compatibility and the cred_t will be fixed size.

  DONE: Needs testing.

- Finish the documentation.

  MOSTLY DONE:
  updated documentation alot.

- Finish the full STREAMS logger and proper implementation of the strlog() utility.

  DONE: Added to strutil package. Needs testing.

- Need to rework the specfs. There are now several situations to consider:

  The following four situations require the specfs.

  a. 2.4 kernel without devfs
  b. 2.6 kernel without devfs
  c. 2.4 kernel with devfs but without devfsd
  d. 2.6 kernel with devfs but without devfsd

  The following two situations could use devfs instead of specfs.

  e. 2.4 kernel with devfs and with devfsd
  f. 2.6 kernel with devfs and with devfsd

  The following one situation could use udev instead of devfs or specfs.

  g. 2.6 kernel with udev

  To get this to work requires that there be an independent layer between the file system providing device access for STREAMS and the STREAMS subsystem. A set of registration functions need to be provided and a common set of call outs from the file system made to the STREAMS executive.

  The registration functions need to be called when a STREAMS driver loads and the file system needs to do the right thing. This also needs to include the registration of major and minor devices, including clone devices.

  The call out functions from the file system need to invoke the STREAMS device file operations in a predictable manner, and the STREAMS subsystem requires the ability to chain open calls, or even open STREAMS devices from within the kernel (e.g. for pipes and connld and such).

  It is difficult to get the file system (specfs, devfs, udev) to hold data structures in a manner that is also usable by the STREAMS subsystem, so the file system adaptation layer needs to maintain data structures in the same manner for all file systems.

  Well,... After a little investigation, it is all messed up. udev doesn't do what we need when demand loading pseudo devices, and devfs is probably not used anymore (I found most production kernels disabled for devfs) so it looks like spefs is the way to go. I might use udev for "real" device drivers, but that's just for SS7. So it looks like we are stuck with mounting the specfs. I notice that ptys still use their own file system too...

  So, what we need now is to rework data structures and the specfs to be a little more stable.

DONE: Tested.

– Kernel objects are another thing. For 2.6 kernels, we need to hold our data structures in the kobject manner so that the /sys file system is usable. This requires another adaptation layer because 2.4 kernels do this in a completely different way. Much of our /proc file system stuff needs to move into /sys for 2.6 kernels by stay the same for 2.4 kernels.

  The /sys file system does not really do much for STREAMS. The /dev/streams specfs file system does more for us.

  SKIPPED.

7.7 History

For the latest developments with regard to history of changes, please see the '`ChangeLog`' file in the release package.

8 Installation

8.1 Repositories

The Linux Fast-STREAMS package release can be accessed from the repositories of The OpenSS7 Project. For `rpm(1)` based systems, the package is available in a `yum(8)` repository based on '`repomd`' XML and may also be accessed using `zypper(8)` or `yast(8)`. For `dpkg(1)` based systems, the package is available in a `apt(8)` repository.

By far the easiest (most repeatable and manageable) form for installing and using *OpenSS7* packages is to install packages from the `yum(8)` or `apt(8)` repositories. If your distribution does not support `yum(8)`, `zypper(8)`, `yast(8)` or `apt(8)`, then it is still possible to install the RPMs or DEBs from the repositories using `rpm(1)`, `dpkg(1)`; or by using `wget(1)` and then installing them from RPM or DEB using `rpm(1)` or `dpkg(1)` locally.

If binaries are not available for your distribution or specific kernel, but your distribution supports `rpm(1)` or `dpkg(1)`, the next best method for installing and using *OpenSS7* packages is to download and rebuild the source RPMs or DSCs from the repository. This can also be performed with `yum(8)`, `zypper(8)`, `yast(8)`, `apt(8)`; or directly using `wget(1)`, `rpm(1)` or `dpkg(1)`.

If your architecture does not support `rpm(1)` or `dpkg(1)` at all, or you have special needs (such as cross-compiling for embedded targets), the final resort method is to download, configure, build and install from tarball. In this later case, the easiest way to build and install *OpenSS7* packages from tarball is to use the tarball for the *OpenSS7 Master Package*, `openss7-0.9.2.G`.

8.1.1 Repositories for YUM

To install or upgrade from the *OpenSS7* '`repomd`' repositories, you will need a file in your '`/etc/yum.repo.d/`' directory. This file can be obtained directly from the *OpenSS7* repository, like so:

```
$> REPOS="http://www.openss7.org/repos/rpms"
$> wget $REPOS/centos/5.2/x86_64/repodata/openss7.repo
$> sudo cp -f openss7.repo /etc/yum.repo.d/
$> sudo yum makecache
```

This example assumes the the distribution is '`centos`' and the distribution release is '`5.2`' and the architecture requires is '`x86_64`'. Another example would be '`$REPOS/i686/suse/11.0/i686/repodata/openss7.repo`', for using `yum(8)` with SUSE.

Once the repository is set up, *OpenSS7* includes a number of virtual package definitions that eas the installation and removal of kernel modules, libraries and utilities. Downloading, configuring, building and installation for a single-kernel distribution is as easy as:

```
$> sudo yum install streams
```

Removing the package is as easy as:

```
$> sudo yum remove streams
```

If you have difficulty downloading the '`openss7.repo`' file, edit the following information into the file and place it into the '`/etc/yum.repo.d/openss7.repo`' file:

```
-| [openss7]
-| enabled = 1
-| name = OpenSS7 Repository
-| baseurl = http://www.openss7.org/repos/rpms/centos/5.2/x86_64
-| gpgcheck = 1
-| gpgkey = http://www.openss7.org/pubkey.asc
```

Note that it is also possible to point to these repositories as an additional installation source when installing CentOS, RedHat, Fedora, or others. You will have an additional *STREAMS* category from which to choose installation packages.

Some additional installation real or virtual package names and the installations they accomplish are as follows:

‘`streams`’
: This package can be used to install or remove the entire Linux Fast-STREAMS package. When installing, kernel modules will be installed automatically for the highest version kernel on your system. When removing, all corresponding kernel modules will also be removed.

‘`streams-devel`’
: This package can be used to install or remove the development components of the Linux Fast-STREAMS package. When installing, ‘`streams`’ and appropriate kernel module and kernel module development and debug packages will also be installed. When removing, the development package and all kernel module development and debug packages will also be removed.

‘`streams-2.4.20-28.7`’
: This package can be used to install or remove the package for a specific kernel version. When installing, the ‘`streams`’ package will also be installed if necessary. When removing the last kernel module package, the ‘`streams`’ package will also be removed.

  Note that the version ‘`2.4.20-28.7`’ is just an example. Use the version returned by ‘`$(uname -r)`’ for the kernel for which you wish to install or remove the packages.

‘`streams-2.4.20-28.7-devel`’
: This package can be used to install or remove the development and debug packages for a specific kernel version. When installing, the ‘`streams`’ and ‘`streams-devel`’ packages will also be installed if necessary. When removing the development and debug for kernel modules for the last kernel, the ‘`streams-devel`’ package will also be removed.

  Note that the version ‘`2.4.20-28.7`’ is just an example. Use the version returned by ‘`$(uname -r)`’ for the kernel for which you wish to install or remove the packages.

For assistance with specific RPMs, see Section 8.2.3 [Downloading the Binary RPM], page 89.

8.1.2 Repositories for APT

For assistance with specific DEBs, see Section 8.2.4 [Downloading the Debian DEB], page 91.

8.2 Downloading

The Linux Fast-STREAMS package releases can be downloaded from the downloads page of The OpenSS7 Project. The package is available as a binary RPM (for popular architectures) a source RPM, Debian binary DEB and source DSC, or as a tar ball. If you are using a browsable viewer, you can obtain the OpenSS7 release of `streams` from the links in the sections that follow.

By far the easiest (most repeatable and manageable) form for installing and using *OpenSS7* packages is to download and install individual packages from binary RPM or DEB. If binary RPMs or DEBs are not available for your distribution, but your distribution supports `rpm(1)` or `dpkg(1)`, the next best method for installing and using *OpenSS7* packages is to download and rebuild the source RPMs or DSCs.

If your architecture does not support `rpm(1)` or `dpkg(1)` at all, or you have special needs (such as cross-compiling for embedded targets), the final resort method is to download, configure, build and install from tarball. In this later case, the easiest way to build and install *OpenSS7* packages from tarball is to use the tarball for the *OpenSS7 Master Package*, `openss7-0.9.2.G`.

8.2.1 Downloading with YUM

OpenSS7 repositories support `yum(8)` and `zypper(8)` in repomd XML format as well as YaST and YaST2 formats.

OpenSS7 includes virtual packages that ease the installation and removal of kernel modules, libraries and utilities. Downloading, configuration, building and installation for a signle-kernel distribution installation is as easy as:

```
% sudo yum install streams
```

This and additional packages for installation are detailed as follows:

‘`streams`’ Install this package if you need the runtime ‘`streams`’ package.

```
% sudo yum install streams
```

This will install the ‘`streams`’, ‘`streams-lib`’ and ‘`streams-KVERSION`’ RPMs, where ‘`KVERSION`’ is the highest version number kernel on your system.

Remove this package if you need to remove all vestages of the ‘`streams`’ package.

```
% sudo yum remove streams
```

This will remove the ‘`streams`’, ‘`streams-lib`’, ‘`streams-devel`’, ‘`streams-KVERSION`’ and ‘`streams-devel-KVERSION`’ RPMs for all kernels on your system.

‘`streams-devel`’

Install this package if you need the development ‘`streams`’ package.

```
% sudo yum install streams-devel
```

This will install the ‘`streams`’, ‘`streams-lib`’, ‘`streams-devel`’, ‘`streams-KVERSION`’ and ‘`streams-devel-KVERSION`’ RPMs, where ‘`KVERSION`’ is the highest version number kernel on your system.

Remove this package if you do not need development capabilities for the ‘`streams`’ package for any kernel.

```
% sudo yum remove streams-devel
```

This will remove the '`streams-devel`' and '`streams-devel-KVERSION`' RPMs for all kernels on your system.

'`streams-2.4.20-28.7`'

Install this package if you need the runtime '`streams`' for kernel version '`2.4.20-28.7`'. The value '`2.4.20-28.7`' is just an example. For the running kernel, you can install the runtime '`streams`' components with:

```
% sudo yum install streams-$(uname -r)
```

This will install the '`streams`', '`streams-lib`' and '`streams-2.4.20-28.7`' RPMs, where '`2.4.20-28.7`' is the kernel version specified.

Remove this package if you no longer need the runtime '`streams`' for kernel version '`2.4.20-28.7`'. The value '`2.4.20-28.7`' is just an example. For the running kernel, you can remove the runtime '`streams`' components with:

```
% sudo yum remove streams-$(uname -r)
```

This will remove the '`streams-2.4.20-28.7`' and '`streams-devel-2.4.20-28.7`' RPMs, where '`2.4.20-28.7`' is the kernel version specified. Also, if this is the last kernel for which '`streams`' was installed, the '`streams`' '`streams-lib`' and '`streams-devel`' RPMs will also be removed.

Note that this is a virtual package name: the actual RPMs installed or removed from the system is a kernel module package whose precise name will depend upon the system being used.

'`streams-devel-2.4.20-28.7`'

Install this package if you need the development '`streams`' package for kernel version '`2.4.20-28.7`'. The value '`2.4.20-28.7`' is just an example. For the running kernel, you can install the kernel development '`streams`' components with:

```
% sudo yum install streams-devel-$(uname -r)
```

This will install the '`streams`', '`streams-lib`', '`streams-devel`', '`streams-2.4.20-28.7`' and '`streams-devel-2.4.20-28.7`' RPMs, where '`2.4.20-28.7`' is the kernel version specified.

Remove this package if you no longer need the development capabilities for the '`streams`' package for kernel version '`2.4.20-28.7`'. The value '`2.4.20-28.7`' is just an example. For the running kernel, you can remove the kernel development '`streams`' components with:

```
% sudo yum remove streams-devel-$(uname -r)
```

This will remove the '`streams-devel-2.4.20-28.7`' RPMs, where '`2.4.20-28.7`' is the kernel version specified. Also, if this is the last kernel for which '`streams`' was installed, the '`streams-devel`' RPMs will also be removed.

Note that this is a virtual package name: the actual RPMs installed or removed from the system is a kernel module package whose precise name will depend upon the system being used.

‘`streams-lib`’
: This package is an auxillary package that should be removed and inserted automatically by `yum(8)`. In rare instances you might need to remove or install this package explicitly.

8.2.2 Downloading with APT

OpenSS7 repositries support `apt(8)` repositorie digests and signatures.

8.2.3 Downloading the Binary RPM

To install from binary RPM, you will need several of the RPM for a complete installation. Binary RPM fall into several categories. To download and install a complete package requires the appropriate RPM from each of the several categories below, as applicable. Some release packages do not provide RPMs in each of the several categories.

To install from Binary RPM, you will need all of the following kernel independent packages for your architecture, and one of the kernel-dependent packages from the next section.

Independent RPM

Independent RPM are not dependent on the Linux kernel version. For example, the source package ‘`streams-source-0.9.2.4-1.7.2.noarch.rpm`’, is not dependent on kernel.

All of the following kernel independent RPM are required for your architecture. Binary RPMs listed here are for example only: additional binary RPMs are available from the downloads site. If your architecture is not available, you can build binary RPM from the source RPM (see see Section 8.4.1 [Building from the Source RPM], page 116).

Architecture Independent

streams-dev-0.9.2.4-1.7.2.noarch.rpm
: The ‘`streams-dev`’ package contains the device definitions necessary to run applications programs developed for Linux Fast-STREAMS.[1]

streams-doc-0.9.2.4-1.7.2.noarch.rpm
: The ‘`streams-doc`’ package contains this manual in plain text, postscript, ‘`pdf`’ and ‘`html`’ forms, along with the meta-information from the ‘`streams`’ package. It also contains all of the manual pages necessary for developing Linux Fast-STREAMS applications and Linux Fast-STREAMS *STREAMS* modules or drivers.

streams-init-0.9.2.4-1.7.2.noarch.rpm
: The ‘`streams-init`’ package contains the `init` scripts and provides the ‘`postinst`’ scripts necessary to create kernel module preloads and modules definitions for all kernel module ‘`core`’ subpackages.

[1] Not all distributions support the ‘`%dev`’ RPM macro: a case in point is the SuSE 8.0 distribution which uses an older version of `rpm(1)`. Distributions that do not support the ‘`%dev`’ macro will build devices as a ‘`%post`’ operation. Note also that not all release packages contain devices. Only packages that provide *STREAMS* character device drivers need devices, and then only when the ‘`specfs`’ or ‘`devfsd`’ is not being used.

streams-source-0.9.2.4-1.7.2.noarch.rpm
: The '`streams-source`' package contains the source code necessary for building the Linux Fast-STREAMS release. It includes the `autoconf(1)` configuration utilities necessary to create and distribute tarballs, '`rpm`' and '`deb`'/'`dsc`'.

Architecture Dependent

streams-devel-0.9.2.4-1.7.2.i686.rpm
: The '`streams-devel`' package contains library archives for static compilation, header files to develop Linux Fast-STREAMS modules and drivers. This also includes the header files and static libraries required to compile Linux Fast-STREAMS applications programs.

streams-lib-0.9.2.4-1.7.2.i686.rpm
: The '`streams-lib`' package contains the run-time shared libraries necessary to run application programs and utilities developed for the '`streams`' package.

streams-util-0.9.2.4-1.7.2.i686.rpm
: The '`streams-util`' package provides administrative and configuration test utilities and commands associated with the Linux Fast-STREAMS package.

Kernel-Dependent RPM

Kernel-Dependent RPM are dependent on specific Linux Kernel Binary RPM releases. Packages are provided for popular released *RedHat* kernels. Packages dependent upon *RedHat* or other kernel RPM will have the '`_kversion`' kernel package version in the package name.

One of the following Kernel-Dependent packages is required for your architecture and kernel version. If your architecture or kernel version is not on the list, you can build binary RPM from the source RPM (see see Section 8.4.1 [Building from the Source RPM], page 116).[2]

streams-core-2.4.20-28.7-0.9.2.4-1.7.2.i686.rpm
: The '`streams-core`' package contains the loadable kernel modules that depend only on the kernel. This package is heavily tied to the kernel for which it was compiled. This particular package applies to kernel version '`2.4.20-28.7`'.[3]

streams-info-2.4.20-28.7-0.9.2.4-1.7.2.i686.rpm
: The '`streams-info`' package[4] contains the module symbol version information for the '`core`' subpackage, above. It is possible to load this subpackage and compile modules that use the exported symbols without loading the actual kernel modules (from the '`core`' subpackage above). This package is heavily tied to the kernel for which it was compiled. This particular package applies to kernel version '`2.4.20-28.7`'.[5]

[2] Note that on *Mandrakelinux*, unlike other RPM kernel distributions, kernel packages for the ix86 architectures are always placed in i586 architecture packages regardless of the true processor architecture of the kernel package. '`configure`' detects this and builds the appropriate packages.

[3] Note that the '`_kversion`' of '`2.4.20-28.7`' is only an example. Note also that only release packages that contain kernel modules will contain a '`core`' subpackage.

[4] Note that only release packages that contain kernel modules and that export versioned symbols will contain a '`info`' subpackage. Also, this subpackage is only applicable to 2.4 series kernels and is not necessary and not built for 2.6 series kernels.

[5] Note that the '`_kversion`' of '`2.4.20-28.7`' is only an example.

Configuration and Installation

To configure, build and install the binary RPM, See Section 8.3.1 [Configuring the Binary RPM], page 97.

8.2.4 Downloading the Debian DEB

To install from binary DEB, you will need several of the DEB for a complete installation. Binary DEB fall into several categories. To download and install a complete package requires the appropriate DEB from each of the several categories below, as applicable. Some release packages do not provide DEBs in each of the several categories.

To install from Binary DEB, you will need all of the following kernel independent packages for your architecture, and one of the kernel-dependent packages from the next section.

Independent DEB

Independent DEB are not dependent on the Linux kernel version. For example, the source package '`streams-source_0.9.2.4-0_i386.deb`', is not dependent on kernel.

All of the following kernel independent DEB are required for your architecture. Binary DEBs listed here are for example only: additional binary DEBs are available from the downloads site. If your architecture is not available, you can build binary DEB from the Debian DSC (see see Section 8.4.2 [Building from the Debian DSC], page 117).

Architecture Independent

streams-dev_0.9.2.4-0_all.deb
: The '`streams-dev`' package contains the device definitions necessary to run applications programs developed for Linux Fast-STREAMS.[6]

streams-doc_0.9.2.4-0_all.deb
: The '`streams-doc`' package contains this manual in plain text, postscript, '`pdf`' and '`html`' forms, along with the meta-information from the '`streams`' package. It also contains all of the manual pages necessary for developing Linux Fast-STREAMS applications and Linux Fast-STREAMS *STREAMS* modules or drivers.

streams-init_0.9.2.4-0_all.deb
: The '`streams-init`' package contains the `init` scripts and provides the postinst scripts necessary to create kernel module preloads and modules definitions for all kernel module '`core`' subpackages.

streams-source_0.9.2.4-0_all.deb
: The '`streams-source`' package contains the source code necessary for building the Linux Fast-STREAMS release. It includes the `autoconf(1)` configuration utilities necessary to create and distribute tarballs, rpms and deb/dscs. !ignore[7] !end ignore

[6] Note that not all release packages contain devices. Only packages that provide *STREAMS* character device drivers need devices, and then only when the '`specfs`' or '`devfsd`' is not being used.

[7] Note that not all releases have source DEB packages. Release packages that do not contain kernel modules do not generate a source DEB package.

Architecture Dependent

streams-devel_0.9.2.4-0_i386.deb
: The '`streams-devel`' package contains library archives for static compilation, header files to develop Linux Fast-STREAMS modules and drivers. This also includes the header files and static libraries required to compile Linux Fast-STREAMS applications programs.

streams-lib_0.9.2.4-0_i386.deb
: The '`streams-lib`' package contains the run-time shared libraries necessary to run application programs and utilities developed for the '`streams`' package.

Kernel-Dependent DEB

Kernel-Dependent DEB are dependent on specific Linux Kernel Binary DEB releases. Packages are provided for popular released *Debian* kernels. Packages dependent upon *Debian* or other kernel DEB will have the '`_kversion`' kernel package version in the package name.

One of the following Kernel-Dependent packages is required for your architecture and kernel version. If your architecture or kernel version is not on the list, you can build binary DEB from the source DEB (see see Section 8.4.2 [Building from the Debian DSC], page 117).[8]

streams-core-2.4.20-28.7_0.9.2.4-0_i386.deb
: The '`streams-core`' package contains the loadable kernel modules that depend only on the kernel. This package is heavily tied to the kernel for which it was compiled. This particular package applies to kernel version '`2.4.20-28.7`'.[9]

streams-info-2.4.20-28.7_0.9.2.4-0_i386.deb
: The '`streams-info`' package[10] contains the module symbol version information for the '`core`' subpackage, above. It is possible to load this subpackage and compile modules that use the exported symbols without loading the actual kernel modules (from the '`core`' subpackage above). This package is heavily tied to the kernel for which it was compiled. This particular package applies to kernel version '`2.4.20-28.7`'.[11]

Configuration and Installation

To configure, build and install the Debian DEB, See Section 8.3.2 [Configuring the Debian DEB], page 98.

8.2.5 Downloading the Source RPM

If you cannot obtain a binary RPM for your architecture, or would like to roll you own binary RPM, download the following source RPM.

---

[8] Note that on *Mandrakelinux*, unlike other DEB kernel distributions, kernel packages for the ix86 architectures are always placed in i586 architecture packages regardless of the true processor architecture of the kernel package. '`configure`' detects this and builds the appropriate packages.

[9] Note that the '`_kversion`' of '`2.4.20-28.7`' is only an example. Note also that only release packages that contain kernel modules will contain a '`core`' subpackage.

[10] Note that only release packages that contain kernel modules and that export versioned symbols will contain a '`info`' subpackage. Also, this subpackage is only applicable to 2.4 series kernels and is not necessary and not built for 2.6 series kernels.

[11] Note that the '`_kversion`' of '`2.4.20-28.7`' is only an example.

streams-0.9.2.4-1.src.rpm
: This is the source RPM for the package. From this source RPM it is possible to build binary RPM for any supported architecture and for any 2.4 or 2.6 kernel.

Configuration

To configure the source RPM, See Section 8.3.3 [Configuring the Source RPM], page 98.

8.2.6 Downloading the Debian DSC

If you cannot obtain a binary DEB for your architecture, or would like to roll your own DEB, download the following Debian DSC.

streams_0.9.2.4-0.dsc
streams_0.9.2.4-0.tar.gz
: This is the Debian DSC for the package. From this Debian DSC it is possible to build binary DEB for any supported architecture and for any 2.4 or 2.6 kernel.

Configuration

To configure the source RPM, See Section 8.3.4 [Configuring the Debian DSC], page 104.

8.2.7 Downloading the Tar Ball

For non-`rpm(1)` and non-`dpkg(1)` architectures, download the tarball as follows:

streams-0.9.2.4.tar.gz
streams-0.9.2.4.tar.bz2
: These are the `tar(1)` balls for the release. These `tar(1)` balls contain the `autoconf(1)` distribution which includes all the source necessary for building and installing the package. These tarballs will even build Source RPM and Binary RPM on `rpm(1)` architectures and Debian DSC and DEB on `dpkg(1)` architectures.

The tar ball may be downloaded easily with `wget(1)` as follows:

```
% wget http://www.openss7.org/streams-0.9.2.4.tar.bz2
```

or

```
% wget http://www.openss7.org/streams-0.9.2.4.tar.gz
```

Note that you will need an *OpenSS7 Project* user name and password to download release candidates (which are only available to subscribers and sponsors of the *OpenSS7 Project*).

Unpacking the Archive

After downloading one of the tar balls, unpack the archive using one of the following commands:

```
% wget http://www.openss7.org/streams-0.9.2.4.tar.gz
% tar -xzvf streams-0.9.2.4.tar.gz
```

or

```
% wget http://www.openss7.org/streams-0.9.2.4.tar.bz2
% tar -xjvf streams-0.9.2.4.tar.bz2
```

Either will create a subdirectory name '`streams-0.9.2.4`' containing all of the files and subdirectories for the `streams` package.

Configuration

To configure and install the tar ball, See Section 8.3.5 [Configuring the Tar Ball], page 104.

8.2.8 Downloading from CVS

If you are a subscriber or sponsor of The OpenSS7 Project with CVS archive access privileges then you can download release, mid-release or release candidate versions of the '`streams`' package from the project CVS archive.

The Linux Fast-STREAMS package is located in the '`streams`' module of '`/var/cvs`'. For release tag information, see Chapter 7 [Releases], page 61.

To access the archive from the project CVS pserver, use the following commands to check out a version from the archive:

```
% export CVSROOT='-d:pserver:username@cvs.openss7.com:2401/var/cvs'
% cvs login
Password: *********
% cvs co -r streams_0.9.2.4 streams
% cvs logout
```

It is, of course, possible to check out by date or by other criteria. For more information, see `cvs(1)`.

Preparing the CVS Working Directory

Although public releases of the '`streams`' package do not require reconfiguration, creating a configurable directory from the CVS archive requires tools not normally distributed with the other releases.

The build host requires the following GNU tools:

- `m4 1.4.12`
- `autoconf 2.63`
- `automake 1.10.1`
- `libtool 2.2.4`
- `gettext 0.17`
- `flex 2.5.33`
- `bison 2.3`

Most desktop development GNU/Linux distributions wil have these tools; however, some non-development or server-style installations might not and they must be installed separately.[12]

Also, these tools can be acquired from the FSF website in the free software directory, and also at the following locations:

- m4-1.4.12
- autoconf-2.63
- automake-1.10.1
- libtool-2.2.4
- gettext-0.17
- flex-2.5.33
- bison-2.3

It should be stressed that, in particular, the `autoconf(1)`, and `automake(1)`, must be at version releases `2.63` and `1.10.1`. *The versions normally distributed in some mainstream GNU/Linux distributions are, in fact, much older than these versions.*[13] GNU version of these packages configured and installed to default directories will install in '`/usr/local/`' allowing them to coexist with distribution installed versions.

For building documentation, the build host also requires the following documentation tools:

- `gs 6.51` or `ghostscript 6.51`, or newer.
- `tetex 3.0` or `texlive 2007`, or newer.
- `texinfo 4.13a` or newer.
- `transfig 3.2.3d` or newer.
- `imagemagick 5.3.8` or `ImageMagick 5.3.8`, or newer.
- `groff 1.17.2` or newer.
- `gnuplot 3.7` or newer.
- `latex2html 1.62` or newer.

Most desktop GNU/Linux distributions will have these tools; however, some server-style installations (e.g. *Ubuntu*-server, *SLES 9* or *Fedora 6 or 7*) will not and they must be installed separately.[14]

Note that `texinfo 4.12` must not be used as it breaks the build process.

For uncooked manual pages, the entire `groff(1)` package is required on *Debian* and *Ubuntu* systems (the base package does not include `grefer(1)` which is used extensively by uncooked manual pages). The following will get what you need:

---

[12] Older version of bison (2.0) and the older version of flex (2.5.4a) are also suitable. Where possible, use the more recent bison 2.3 and flex 2.5.33.

[13] A notable exception is Debian and Fedora 7. Note that on Fedora 7 the `gettext-devel` package must be installed.

[14] In particular, for *CentOS*, *Fedora 6 or 7*, the `tetex-latex` and `gnuplot` packages must be loaded as well. Note also that the `latex2html` used to be part of the `textex` package (or subpackages) but is now often packaged on its own. Recent distributions such as SUSE 11.0 and Fedora 9 use the `texlive` package instead of the `tetex` package.

```
Debian: % apt-get install groff_ext
Ubuntu: % apt-get install groff
```

In addition, the build host requires a complete tool chain for compiling for the target host, including kernel tools such as genksyms(8) and others.

If you wish to package 'rpms' on an rpm(1) system, or 'debs' on a dpkg(1) system, you will need the appropriate tool chain. Systems based on rpm(1) typically have the necessary tool chain available, however, dpkg(1) systems do not. The following on a *Debian* or *Ubuntu* system will get what you need:

```
% apt-get install debhelper
% apt-get install fakeroot
```

To generate a configuration script and the necessary scriptlets required by the GNU autoconf(1) system, execute the following commands on the working directory:

```
% autoreconf -fiv streams
```

where, 'streams' is the name of the directory to where the working copy was checked out under the previous step. This command generates the 'configure' script and other missing pieces that are normally distributed with the release Tar Balls, SRPMs and DSCs.

Make sure that 'autoreconf --version' returns '2.63'. Otherwise, you may need to perform something like the following:

```
% PATH="/usr/local/bin:$PATH"
% autoreconf -fiv streams
```

After reconfiguring the directory, the package can then be configured and built using the same instructions as are used for the Tar Ball, see Section 8.3.5 [Configuring the Tar Ball], page 104, and Section 8.4.3 [Building from the Tar Ball], page 117.

Do note, however, that make(1) will rebuild the documentation that is normally released with the package. Additional tools may be necessary for building the documentation. To avoid building and installing the documentation, use the '--disable-devel' or '--disable-docs' option to configure described in Section 8.3.5 [Configuring the Tar Ball], page 104.

When configuring the package in a working directory and while working a change-compile-test cycle that involves configuration macros or documentation, I find it of great advantage to invoke the GNU 'configure' options '--enable-maintainer-mode', '--enable-dependency-tracking' and '--disable-devel'. The first of these three options will add maintainer-specific targets to any generated 'Makefile', the second option will invoke automatic dependency tracking within the 'Makefile' so rebuilds after changes to macro, source or documentation files will be automatically rebuilt; and the last option will suppress rebuilding and reinstalling documentation manual pages and header files. Header files will still be available under the '/usr/src' directory.

8.3 Configuration

8.3.1 Configuring the Binary RPM

In general the binary RPM do not require any configuration, however, during installation it is possible to relocate some of the installation directories. This allows some degree of customization. Relocations that are available on the binary RPM are as follows:

‘`streams-core-2.4.20-28.7-0.9.2.4-1.7.2.i686.rpm`’

‘`/lib/modules/2.4.20-28.7`’
: This relocatable directory contains the kernel modules that provide the streams core, drivers and modules.[15]

‘`streams-info-2.4.20-28.7-0.9.2.4-1.7.2.i686.rpm`’

‘`/usr/include/streams/2.4.20-28.7`’
: This relocatable directory contains the kernel module exported symbol information that allows other kernel modules to be compiled against the correct version of the streams package.[16]

‘`streams-dev-0.9.2.4-1.7.2.i686.rpm`’
: (not relocatable)

‘`streams-devel-0.9.2.4-1.7.2.i686.rpm`’

‘`/usr/lib`’
: This relocatable directory contains streams libraries.

‘`/usr/include/streams`’
: This relocatable directory contains streams header files.

‘`streams-doc-0.9.2.4-1.7.2.i686.rpm`’

‘`/usr/share/doc`’
: This relocatable directory contains all package specific documentation (including this manual). The subdirectory in this directory is the ‘`streams-0.9.2.4`’ directory.

‘`/usr/share/info`’
: This relocatable directory contains info files (including the info version of this manual).

‘`/usr/share/man`’
: This relocatable directory contains manual pages.

‘`streams-lib-0.9.2.4-1.7.2.i686.rpm`’

‘`/usr/lib`’
: This relocatable directory contains the run-time shared libraries necessary to run applications programs and utilities developed for Linux Fast-STREAMS.

---

[15] Note that the ‘`_kversion`’ of ‘`2.4.20-28.7`’ is only an example.

[16] Note that the ‘`_kversion`’ of ‘`2.4.20-28.7`’ is only an example. Also, note that the ‘`info`’ subpackage is only applicable to the 2.4 kernel series.

‘`/usr/share/locale`’
: This relocatable directory contains the locale information for shared library files.

‘`streams-source-0.9.2.4-1.7.2.i686.rpm`’

‘`/usr/src`’
: This relocatable directory contains the source code.

‘`streams-util-0.9.2.4-1.7.2.i686.rpm`’

‘`/usr/bin`’
: This relocatable directory contains binary programs and utilities.

‘`/usr/sbin`’
: This relocatable directory contains system binary programs and utilities.

‘`/usr/libexec`’
: This relocatable directory contains test programs.

‘`/etc`’ This relocatable directory contains `init` scripts and configuration information.

Installation

To install the binary RPM, See Section 8.5.1 [Installing the Binary RPM], page 118.

8.3.2 Configuring the Debian DEB

In general the binary DEB do not require any configuration.

Installation

To install the Debian DEB, See Section 8.5.2 [Installing the Debian DEB], page 119.

8.3.3 Configuring the Source RPM

When building from the source RPM (see Section 8.4.1 [Building from the Source RPM], page 116), the rebuild process uses a number of macros from the user’s ‘`.rpmmacros`’ file as described in `rpm(8)`.

Following is an example of the ‘`~/.rpmmacros`’ file that I use for rebuilding RPMS:

```
#
# RPM macros for building rpms
#

%vendor OpenSS7 Corporation
%distribution OpenSS7
%disturl http://www.openss7.org/
%packager Brian Bidulock <bidulock@openss7.org>
%url http://www.openss7.org/

%_signature gpg
%_gpg_path /home/brian/.gnupg
%_gpg_name openss7@openss7.org
%_gpgbin /usr/bin/gpg

%_source_payload w9.bzdio
%_binary_payload w9.bzdio

%_unpackaged_files_terminate_build 1
%_missing_doc_files_terminate_build 1
%_use_internal_dependency_generator 0
%_repackage_all_erasures 0
%_rollback_transaction_on_failure 0

%configure2_5x %configure
%make make
```

When building from the source RPM (see Section 8.4.1 [Building from the Source RPM], page 116), it is possible to pass a number of additional configuration options to the `rpmbuild(1)` process.

The additional configuration options are described below.

Note that distributions that use older versions of rpm do not have the '`--with`' or '`--without`' options defined. To achieve the same effect as:

```
--with someparm=somearg
```

do:

```
--define "_with_someparm --with-someparm=somearg"
```

`--define "_kversion $PACKAGE_KVERSION"`
: Specifies the kernel version other than the running kernel for which to build. If `_kversion` is not defined when rebuilding, the environment variable *PACKAGE_KVERSION* is used. If the environment variable *PACKAGE_KVERSION* is not defined, then the version of the running kernel (i.e. discovered with '`uname -r`') is used as the target version for kernel-dependent packages. This option can also be defined in an '`.rpmspec`' file using the macro name '`_kversion`'.

`--with checks`
`--without checks`
: Enable or disable preinstall checks. Each packages supports a number of preinstall checks that can be performed by invoking the '`check`' target with `automake(1)`. These currently consist of checking each kernel module for unresolved kernel symbols, checking for documentation for exported kernel

module symbols, checking for documentation for exported library symbols, checking for standard options for build and installable programs, checking for documentation for built and installable programs. Normally these checks are only run in maintainer mode, but can be enabled and disabled with this option.

`--with k-optimize=HOW`
`--without k-optimize`
: Specify '`HOW`' optimization, *normal*, *size*, *speed* or *quick*. *size* compiles kernel modules `-Os`, *speed* compiles kernel modules `-O3`, and *quick* compiles kernel modules `-O0`. The default is *normal*. Use with care.

`--with cooked-manpages`
`--without cooked-manpages`
: Some systems do not like `grefer(1)` references in manual pages.[17] This option will cook `soelim(1)`, `refer(1)`, `tbl(1)` and `pic(1)` commands from the manual pages and also strip `groff(1)` comments. The default is to leave manual pages uncooked: they are actually smaller that way.

`--with public`
`--without public`
: Release public packages or private packages. This option has no effect on the '`streams`' package. The default is to release public packages.

`--with k-debug`
`--without k-debug`
: Specifies whether kernel debugging is to be performed on the build kernel modules. Mutually exclusive with `test` and `safe` below. This has the effect of removing static and inline attributes from functions and invoking all debugging macros in the code. The default is to not perform kernel debugging.

`--with k-test`
`--without k-test`
: Specifies whether kernel testing is to be performed. Mutually exclusive with `debug` above and `safe` below. This has the effect of removing static and inline attributes from functions and invoking most debugging macros in the code. The default is to not perform kernel testing.

`--with k-safe`
`--without k-safe`
: Specifies whether kernel saftey is to be performed. Mutually exclusive with `debug` and `test` above. This has the effect of invoking some more pedantic assertion macros in the code. The default is not to apply kernel safety.

`--with k-inline`
`--without k-inline`
: Specifies whether kernel `inline` functions are to be placed inline. This has the effect of adding the '`-finline-functions`' flag to *CFLAGS* for compil-

[17] In particular, some *Debian* systems do not load the `groff(1)` extensions package and do not have `grefer(1)` installed. Although this is an oversight on the configuration of the particular *Debian* system, we accomodate such misconfiguration with this feature.

ing kernel modules. Linux 2.4 kernels are normally compiled '`-O2`' which does not respect the `inline` directive. This compiles kernel modules with '`-finline-functions`' to get closer to '`-O3`' optimization. For better optimization controls, See Section 8.3.5 [Configuring the Tar Ball], page 104.

`--with k-modversions`
`--without k-modversions`
Specifies whether kernel symbol versions are to be applied to symbols exported by package kernel modules. The default is to version exported module symbols. This package does not export symbols so this option has no effect.

`--with devfs`
`--without devfs`
Specifies whether the build is for a device file system daemon enabled system with autoloading, or not. The default is to build for `devfsd(1)` autoloading when *CONFIG_DEVFS_FS* is defined in the target kernel. The '`rebuild`' target uses this option to signal to the RPM spec file that the '`dev`' subpackage need not be built. This option does not appear when the package has no devices.

`--with devel`
`--without devel`
Specifies whether to build development environment packages such as those that include header files, static libraries, manual pages and `texinfo(1)` documentation. The default is to build development environment packages. This option can be useful when building for an embedded target where only the runtime components are desired.

`--with docs`
`--without docs`
Specifies whether to build and install major documentation such manual pages and `texinfo(1)` documentation. The default is to build and install documentation. This option can be useful when building for an embedded target where only the runtime and static compile components are desired, but not major documentation. This option does not override the setting of `--without devel`.

`--with tools`
`--without tools`
Specifies whether user space packages are to be built. The default is to build user space packages. This option can be useful when rebuilding for multiple architectures and target kernels. The '`rebuild`' `automake(1)` target uses this feature when rebuilding for all available architectures and kernels, to rebuild user packages once per architecture instead of once per kernel.

`--with modules`
`--without modules`
Specifies whether kernel modules packages are to be built. The default is to build kernel module packages. This option can be useful when rebuilding for multiple architectures and target kernels. The '`rebuild`' `automake(1)` target uses this feature to rebuild for all available architectures and kernels.

In addition, the following `rpm` options, specific to the Linux Fast-STREAMS package are available:

`--with streams-syncqs`
: When enabled, MP synchronization queues are enabled for SMP kernels. This option defaults to '`disabled`'. This option is not tested for early releases.

`--without streams-kthreads`
: When enabled, the STREAMS scheduler runs as a kernel thread. When disabled, the STREAMS scheduler runs as a software interrupt (bottom half). Running the STREAMS scheduler at bottom half instead of a kernel thread breaks the '`strinet`' driver, which must be able to invoke kernel functions that might sleep (but don't). This option defaults to '`enabled`'.

`--without streams-utils`
: I have experimented with putting the STREAMS utilities into their own package, '`strutil`', however, this is not complete yet. This option defaults to '`enabled`'. Do not disable this option.

`--without big-compile`
: When enabled, the STREAMS, the Stream head, and the clone driver are all compiled together in one big compilation unit. This allows the compiler greater opportunity to optimize. This option defaults to '`enabled`'. Do not disable this option.

`--with module-sth`
: Enable '`sth`' (stream head) module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module, unless option '`big-compile`' is specified. This option defaults to '`disabled`'. This option defaults to '`enabled`' if '`big-compile`' is enabled.

`--with module-bufmod`
: Enable '`bufmod`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with module-nullmod`
: Enable '`nullmod`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with module-pipemod`
: Enable '`pipemod`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with module-connld`
: Enable '`connld`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with module-sc`
: Enable '`sc`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with module-testmod`
: Enable '`testmod`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-clone`
: Enable '`clone`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module, unless option '`big-compile`' is specified. This option defaults to '`disabled`'. This option defaults to '`enabled`' if '`big-compile`' is enabled.

`--with driver-echo`
: Enable '`echo`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-fifo`
: Enable '`fifo`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-log`
: Enable '`log`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-loop`
: Enable '`loop`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-nsdev`
: Enable '`nsdev`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-mux`
: Enable '`mux`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-nuls`
: Enable '`nuls`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-pipe`
: Enable '`pipe`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-sad`
: Enable '`sad`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-sfx`
: Enable '`sfx`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with driver-spx`
: Enable '`spx`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--with streams-fifos`
: Enable override of system fifos with STREAMS-based fifos. This option defaults to '`disabled`'. This option is not tested. Do not enable this option yet.

`--with streams-bcm`
: Enable STREAMS binary compatibility mode. When enabled, exported functions (and callouts) will never pass three arguments in registers on architectures supporting `regparm` (i.e. *__i386__*, *__x86_64__*, *__k8__*) regardless of how the kernel was compiled. Additional *LiS* binary compatibility is also enabled (e.g. credentials). This option defaults to '`disabled`'.

In general, the default values of these options are sufficient for most purposes and no options need be provided when rebuilding the Source RPMs.

Build

To build from the source RPM, See Section 8.4.1 [Building from the Source RPM], page 116.

8.3.4 Configuring the Debian DSC

The Debian DSC can be configured by passing options in the environment variable *BUILD_DEBOPTIONS*. The options placed in this variable take the same form as those passed to the '`configure`' script, See Section 8.3.5 [Configuring the Tar Ball], page 104. For an example, See Section 8.4.2 [Building from the Debian DSC], page 117.

Build

To build from the Debian DSC, See Section 8.4.2 [Building from the Debian DSC], page 117.

8.3.5 Configuring the Tar Ball

All of the normal GNU `autoconf(1)` configuration options and environment variables apply. Additional options and environment variables are provided to tailor or customize the build and are described below.

8.3.5.1 Configure Options

Following are the additional '`configure`' options, their meaning and use:

`--enable-checks`
`--disable-checks`
: Enable or disable preinstall checks. Each release package supports a number of preinstall checks that can be performed by invoking the '`check`' target with `make(1)`. These currently consist of checking each kernel module for unresolved kernel symbols, checking for documentation for exported kernel module symbols, checking for documentation for exported library symbols, checking for standard options for build and installable programs, checking for documentation for built and installable programs. Normally these checks are only run in maintainer mode, but can be enabled and disabled with this option.

`--enable-autotest`
`--disable-autotest`
: Enable or disable pre- and post-installation testing. Each release package supports a number of `autotest` test suites that can be performed by invoking the '`installcheck`' target with `make(1)`. These currently consist of running installed modules, commands and binaries against a number of specific test cases. Normally these checks are only run in maintainer mode, but can be enabled and disabled with this option.

`--disable-compress-manpages`
: Compress manual pages with '`gzip -9`' or '`bzip2 -9`' or leave them uncompressed. The default is to compress manual pages with '`gzip -9`' or '`bzip2 -9`' if a single compressed manual page exists in the target installation directory ('`--mandir`'). This disables automatic compression.

`--disable-public`
: Disable public release. This option is not usable on public releases and only has a usable effect on Linux Fast-STREAMS when the package is acquired from CVS. In particular, the *STREAMS SS7/VoIP/ISDN/SIGTRAN Stacks* (`strss7-0.9a.8`) release package has a large number of non-public components. Specifying this option will cause the package to build and install all private release components in addition to the public release components. This option affects all release packages. Most release packages do not have private release components.

`--disable-initscripts`
: Disables the installation of `init` scripts. The default is to configure and install `init` scripts and their associated configuration files.

  Although the default is to install `init` scripts, installation attempts to detect a System V `init` script configuration, and if one is not found, the `init` scripts are installed into the appropriate directories, but the symbolic links to the run level script directories are not generated and the script is not invoked. Therefore, it is safe to leave this option unchanged, even on distributions that do not support System V `init` script layout.

`--disable-32bit-libs`
: Disables the build and install of 32-bit compatibility libraries and test binaries on 64-bit systems that support 32-bit compatibility. The default is to build and install 32-bit compatibility libraries and test binaries. This option can be usefule when configuring for an embedded target where only native shared libraries and binaries are desired.

`--disable-devel`
: Disables the installation of development environment components such as header files, static libraries, manual pages and `texinfo(1)` documentation. The default is to install development environment components. This option can be useful when configuring for an embedded target where only the runtime components are desired, or when performing a edit-compile-test cycle.

`--disable-docs`
: Disables the build and installation of major documentation such manual pages and `texinfo(1)` documentation. The default is to build and install documentation. This option can be useful when building for an embedded target where only the runtime and static compile components are desired, but not major documentation. This option does not override the setting of '`--disable-devel`'.

`--enable-tools`
: Specifies whether user space programs and libraries are to be built and installed. The default is to build and install user space programs and libraries. This option can be useful when rebuilding for multiple architectures and target kernels, particularly under `rpm(1)` or `dpkg(1)`. The '`rebuild`' `automake(1)` target uses this feature when rebuilding RPMs for all available architectures and kernels, to rebuild user packages once per architecture instead of once per kernel.

`--enable-modules`
: Specifies whether kernel modules are to be built and installed. The default is to build and install kernel modules. This option can be useful when rebuilding for multiple architectures and target kernels, particularly under `rpm(1)` or `dpkg(1)`. The '`rebuild`' `automake(1)` target uses this feature to rebuild for all available architectures and kernels.

`--enable-arch`
: Specifies whether architectural dependent package components are to be built and installed. This option can be useful when rebuilding for multiple architectures and target kernels, particularly under `dpkg(1)`. The default is to configure, build and install architecture dependent package components.

`--enable-indep`
: Specifies whether architecture independent package components are to be built and installed. This option can be useful when rebuilding for multiple architectures and target kernels, particularly under `dpkg(1)`. The default is to configure, build and install architecture independent package components.

`--enable-k-inline`
: Enable kernel inline functions. Most Linux kernels build without '`-finline-functions`'. This option adds the '`-finline-functions`' and '`-Winline`' flags to the compilation of kernel modules. Use with care.

`--enable-k-safe`
: Enable kernel module run-time safety checks. Specifies whether kernel safety is to be performed. This option is mutually exclusive with '`--enable-k-test`' and '`--enable-k-debug`' below. This has the effect of invoking some more pedantic assertion macros in the code. The default is not to apply kernel safety.

`--enable-k-test`
: Enable kernel module run-time testing. Specifies whether kernel testing is to be performed. This option is mutually exclusive with '`--enable-k-safe`' above and '`--enable-k-debug`' below. This has the effect of remove `static` and `inline` attributes from functions and invoking most non-performance affecting debugging macros in the code. The default is not to perform kernel testing.

`--enable-k-debug`
: Enable kernel module run-time debugging. Specifies whether kernel debugging is to be performed. This option is mutually exclusive with '`--enable-k-safe`' and '`--enable-k-test`' above. This has the effect of removing `static` and `inline` attributes from functions and invoking all debugging macros in the code (including performance-affecting debug macros). The default is to not perform kernel debugging.

`--enable-devfs`
`--disable-devfs`
: Specifies whether the build is for a device file system daemon enabled system with autoloading, or not. The default is to build for `devfsd(8)` autoloading when *CONFIG_DEVFS_FS* is defined in the target kernel. The '`reuild`' `automake(1)` target uses this option to signal to the RPM spec file that the '`dev`' subpackage need not be built. This option has no effect for release packages that do not provide devices.

`--with-gpg-user=GNUPGUSER`
: Specify the `gpg(1)` '`GNUPGUSER`' for signing RPMs and tarballs. The default is the content of the environment variable *GNUPGUSER*. If unspecified, the `gpg(1)` program will normally use the user name of the account invoking the `gpg(1)` program. For building source RPMs, the RPM macro '`_gpg_name`' will override this setting.

`--with-gpg-home=GNUPGHOME`
: Specify the '`GNUPGHOME`' directory for signing RPMs and tarballs. The default is the user's '`~/.gpg`' directory. For building source RPMs, the RPM macro '`_gpg_path`' will override this setting.

`--with-pkg-epoch=EPOCH`
: Specifies the epoch for the package. This is neither used for `rpm(1)` nor `dpkg(1)` packages, it applies to the tarball release as a whole. The default is the contents of the '`.pkgepoch`' file in the release package source directory or, if that file does not exist, zero (0).

`--with-pkg-release=RELEASE`
: Specifies the release for the package. This is neither used for `rpm(1)` nor `dpkg(1)` packages, it applies to the tarball release as a whole. The default is the contents of the '`.pkgrelease`' file in the release package source directory or, if that file does not exist, one (1). This is the number after the last point in the package version number.

`--with-pkg-distdir=DIR`
: Specifies the distribution directory for the package. This is used by the maintainer for building distributions of tarballs. This is the directory into which archives are copied for distribution. The default is the top build directory.

`--with-cooked-manpages`
: Convert manual pages to remove macro dependencies and `grefer(1)` references. Some systems do not like `grefer(1)` references in manual pages.[18] This option will cook `soelim(1)`, `refer(1)`, `tbl(1)` and `pic(1)` commands from the manual pages and also strip `groff(1)` comments. The default is to leave manual pages uncooked (they are actually smaller that way).

`--with-rpm-epoch=PACKAGE_EPOCH`
: Specify the '`PACKAGE_EPOCH`' for the RPM spec file. The default is to use the RPM epoch contained in the release package file '`.rpmepoch`'.

`--with-rpm-release=PACKAGE_RPMRELEASE`
: Specify the '`PACKAGE_RPMRELEASE`' for the RPM '`spec`' file. The default is to use the RPM release contained in the release package file '`.rpmrelease`'.

`--with-rpm-extra=PACKAGE_RPMEXTRA`
: Specify the '`PACKAGE_RPMEXTRA`' extra release information for the RPM spec file. The default is to use the RPM extra release information contained in the release package file '`.rpmextra`'. Otherwise, this value will be determined from automatic detection of the RPM distribution.

`--with-rpm-topdir=PACKAGE_RPMTOPDIR`
: Specify the '`PACKAGE_RPMTOPDIR`' top directory for RPMs. If specified with a null '`PACKAGE_RPMTOPDIR`', the default directory for the RPM distribution will be used. If this option is not provided on the command line, the top build directory will be used as the RPM top directory as well.

`--with-deb-epoch=EPOCH`
: Specify the '`PACKAGE_DEBEPOCH`' for the DEB control file. The default is to use the DEB epoch contained in the release package file '`.debepoch`'.

`--with-deb-release=RELEASE`
: Specify the '`PACKAGE_DEBRELEASE`' for the DEB control file. The default is to use the DEB release contained in the release package file '`.debrelease`'.

`--with-deb-topdir=DIR`
: Specify the '`PACKAGE_DEBTOPDIR`' top directory for DEBs. If specified with a null '`PACKAGE_DEBTOPDIR`', the default directory for the DEB distribution will be used. If this option is not provided on the command line, the top build directory will be used as the DEB top directory as well.

`--with-k-release=PACKAGE_KRELEASE`
: Specify the '`PACKAGE_KRELEASE`' release of the Linux kernel for which the build is targeted. When not cross compiling, if this option is not set, the build will be targeted at the kernel running in the build environment (e.g., '`uname -r`'). When cross-compiling this option must be specified or the configure script will generate an error and terminate.

[18] In particular, some *Debian* or *Ubuntu* systems do not load the `groff(1)` extensions package and do not have `grefer(1)` installed. Although this is an oversight on the configuration of the particular *Debian* or *Ubuntu* system, we accomodate such misconfiguration with this feature.

`--with-k-linkage=PACKAGE_KLINKAGE`
: Specify the '`PACKAGE_KLINKAGE`' for kernel module linkage. This can be one of the following:
  - '`loadable`' – loadable kernel modules
  - '`linkable`' – linkable kernel objects

  The default is to build loadable kernel modules.

`--with-k-modules=K-MODULES-DIR`
: Specify the '`K-MODULES-DIR`' directory to which kernel modules will be installed. The default is based on the option '`--with-k-release`', '`--with-k-prefix`' and '`--with-k-rootdir`'. The default is '`DESTDIR`'/'`K-MODULES-DIR`' which is typically '*`DESTDIR`*`/lib/modules/`*`PACKAGE_KRELEASE`*`/`'. This directory is normally located by the '`configure`' script and need only be provided for special cross-build environments or when requested by a '`configure`' script error message.

`--with-k-build=K-BUILD-DIR`
: Specify the '`K-BUILD-DIR`' base kernel build directory in which configured kernel source resides. The default is '*`DESTDIR`*`/`*`K-MODULES-DIR`*`/build`'. This directory is normally located by the '`configure`' script and need only be provided for special cross-build environments or when requested by a '`configure`' script error message.

`--with-k-source=K-SOURCE-DIR`
: Specify the '`K-SOURCE-DIR`' base kernel build directory in which configured kernel source resides. The default is '*`DESTDIR`*`/`*`K-MODULES-DIR`*`/source`'. This directory is normally located by the '`configure`' script and need only be provided for special cross-build environments or when requested by a '`configure`' script error message.

`--with-k-modver=K-MODVER-FILE`
: Specify the '`K-MODVER-FILE`' kernel module versions file. The default is '*`K-BUILD-DIR`*`/Module.symvers`'. This file is normally located by the '`configure`' script and need only be provided for special cross-build environments or when requested by a '`configure`' script error message.

`--with-k-sysmap=K-SYSMAP-FILE`
: Specify the '`K-SYSMAP-FILE`' kernel system map file. The default is '*`K-BUILD-DIR`*`/System.map`'. This file is normally located by the '`configure`' script and need only be provided for special cross-build environments or when requested by a '`configure`' script error message.

`--with-k-archdir=K-ARCHDIR`
: Specify the '`K-ARCHDIR`' kernel source architecture specific directory. The default is '*`DESTDIR`*`/`*`K-SOURCE-DIR`*`/arch`'. This directory is normally located by the '`configure`' script and need only be provided for special cross-build environments or when requested by a '`configure`' script error message.

`--with-k-machdir=K-MACHDIR`
: Specify the '`K-MACHDIR`' kernel source machine specific directory. The default is '*`DESTDIR`*`/`*`K-SOURCE-DIR`*`/target_cpu`'. This directory is normally located

by the '`configure`' script and need only be provided for special cross-build environments or when requested by a '`configure`' script error message.

`--with-k-config=K-CONFIG`
: Specify the '`K-CONFIG`' kernel configuration file. The default is '*`BOOT`*`/config-`*`K-RELEASE`*'. This configuration file is normally located by the '`configure`' script and need only be provided for special cross-build environments or when requested by a '`configure`' script error message.

`--with-k-optimize=HOW`
`--without-k-optimize`
: Specify '`HOW`' optimization, *normal*, *size*, *speed* or *quick*. *size* compiles kernel modules `-Os`, *speed* compiles kernel modules `-O3`, and *quick* compiles kernel modules `-O0`. The default is *normal*. Use with care. The most common use of this option is to specify '`--with-k-optimize=speed --disable-k-safe`' to compile for maximum performance. Nevertheless, even these setting are *ricing* and the resulting kernel modules will only be about 5% faster.

`--with-strconf-master=STRCONF_CONFIG`
: Specify the '`STRCONF_CONFIG`' file name to which the configuration master file is written. The default is '`Config.master`'.

`--with-base-major=STRCONF_MAJBASE`
: Start numbering for major devices at '`STRCONF_MAJBASE`'. The default is '`230`'.

In addition, the following `configure` options, specific to the Linux Fast-STREAMS package are available:

`--enable-streams-bcm`
: Enable *STREAMS* binary compatibility mode. When enabled, exported functions (and callouts) will never pass three arguments in registers on architectures supporting `regparm` (i.e. *`__i386__`*, *`__x86_64__`*, *`__k8__`*) regardless of how the kernel was compiled. Additional *LiS* binary compatibility is also enabled (e.g. credentials). This option defaults to '`disabled`'.

`--disable-streams-irq`
: Disable *STREAMS* irq suppression. When enabled, hard interrupts are suppressed whenever locks are taken throughout the *STREAMS* executive. When profiling using masked interrupt driven profilers, this makes profiling difficult. When disabled, soft interrupts are suppressed only. This allows interrupt driven profilers to give more meaningful information. Do not disable hard interrupt suppression unless you know what your are doing. Also, as the softirq threads are run whenever soft interrupt locks are released, this has a negative impact on *STREAMS* performance. This option defaults to '`enabled`'.

`--enable-streams-stats`
: Enable *STREAMS* statistics counting. When disabled, it is the module or driver's responsibility to increment *STREAMS* statistics upon entry to putp, srvp, qopen, qclose and qadmin procedures. When enabled, *STREAMS* will perform statistics counting of entry to the putp, srvp, qopen, qclose and qadmin procedures whenever a module_stat structure is associated with the queue.

The scls(8) utility can be used to display the statistics of any module or driver. Enabling this feature has a slight negative impact on performance and is contrary to *SVR 4.2* behaviour and is, therefore, disabled by default. This option defaults to '`disabled`'.

`--enable-streams-syncqs`
When enabled, MP synchronization queues are enabled for SMP kernels. This option defaults to '`disabled`'. This option is not tested for early releases.

`--disable-streams-kthreads`
When enabled, the *STREAMS* scheduler runs as a kernel thread. When disabled, the *STREAMS* scheduler runs as a software interrupt (bottom half). Running the *STREAMS* scheduler at bottom half instead of a kernel thread breaks the '`strinet`' driver, which must be able to invoke kernel functions that might sleep (but don't). This option defaults to '`enabled`'.

`--disable-streams-utils`
We experimented with putting the *STREAMS* utilities into their own package, '`strutil`', however, this is not complete yet. As the '`strutil`' was only necessary because *LiS* did not support these utilities itself, and *LiS* is deprecated, the '`strutil`' package is also deprecated, and this feature should always be left enabled. This option defaults to '`enabled`'. Do not disable this option.

`--disable-big-compile`
When enabled, the *STREAMS*, the Stream head, and the clone driver are all compiled together in one big compilation unit. This allows the compiler greater opportunity to optimize. This option defaults to '`enabled`'. Do not disable this option.

`--enable-module-sth`
Enable '`sth`' (stream head) module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module, unless option '`big-compile`' is specified. This option defaults to '`disabled`'. This option defaults to '`enabled`' if '`big-compile`' is enabled.

`--enable-module-bufmod`
Enable '`bufmod`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-module-nullmod`
Enable '`nullmod`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-module-pipemod`
Enable '`pipemod`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-module-connld`
: Enable '`connld`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-module-sc`
: Enable '`sc`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-module-testmod`
: Enable '`testmod`' module linked into '`streams`' object. The default is to create the module as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-clone`
: Enable '`clone`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module, unless option '`big-compile`' is specified. This option defaults to '`disabled`'. This option defaults to '`enabled`' if '`big-compile`' is enabled.

`--enable-driver-echo`
: Enable '`echo`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-fifo`
: Enable '`fifo`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-log`
: Enable '`log`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-loop`
: Enable '`loop`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-nsdev`
: Enable '`nsdev`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-mux`
: Enable '`mux`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-nuls`
: Enable '`nuls`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-pipe`
: Enable '`pipe`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-sad`
: Enable '`sad`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-sfx`
: Enable '`sfx`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-driver-spx`
: Enable '`spx`' driver linked into '`streams`' object. The default is to create the driver as a separate loadable kernel module. This option defaults to '`disabled`'.

`--enable-streams-fifos`
: Enable override of system fifos with *STREAMS*-based fifos. This option defaults to '`disabled`'. This option is not tested. Do not enable this option yet.

8.3.5.2 Environment Variables

Following are additional environment variables to '`configure`', their meaning and use:

GPG
: GPG signature command. This is used for signing distributions by the maintainer. By default, '`configure`' will search for this tool.

GNUPGUSER
: GPG user name. This is used for signing distributions by the maintainer.

GNUPGHOME
: GPG home directory. This is used for signing distributions by the maintainer.

GPGPASSWD
: GPG password for signing. This is used for signing distributions by the maintainer. This environment variable is not maintained by the '`configure`' script and should only be used on an isolated system.

SOELIM
: Roff source elimination command, `soelim(1)`. This is only necessary when the option '`--with-cooked-manpages`' has been specified and '`configure`' cannot find the proper `soelim(1)` command. By default, '`configure`' will search for this tool.

REFER
: Roff references command, `refer(1)`. This is only necessary when the option '`--with-cooked-manpages`' has been specified and '`configure`' cannot find the proper `refer(1)` command. By default, '`configure`' will search for this tool.

TBL
: Roff table command, `tbl(1)`. This is only necessary when the option '`--with-cooked-manpages`' has been specified and '`configure`' cannot find the proper `tbl(1)` command. By default, '`configure`' will search for this tool.

PIC
: Roff picture command, `pic(1)`. This is only necessary when the option '`--with-cooked-manpages`' has been specified and '`configure`' cannot find the proper `pic(1)` command. By default, '`configure`' will search for this tool.

GZIP
: Default compression options provided to `GZIP_CMD`.

GZIP_CMD
: Manpages (and kernel modules) compression commands, `gzip(1)`. This is only necessary when the option '`--without-compressed-manpages`' has *not* been

specified and '`configure`' cannot find the proper `gzip(1)` command. By default, '`configure`' will search for this tool.

BZIP2 Default compression options provided to `BZIP2_CMD`

BZIP2_CMD
Manpages compression commands, `bzip2(1)`. This is only necessary when the option '`--without-compressed-manpages`' has *not* been specified and '`configure`' cannot find the proper `bzip2(1)` command. By default, '`configure`' will search for this tool.

MAKEWHATIS
Manpages apropros database rebuild command, `makewhatis(8)`. By default, '`configure`' will search for this tool. By default, '`configure`' will search for this tool.

CHKCONFIG
Chkconfig command, `chkconfig(8)`. This was used for installation of `init` scripts. All packages now come with `init_install(8)` and `init_remove(8)` scripts used to install and remove `init` scripts on both RPM and Debian systems.

RPM Rpm command, `rpm(1)`. This is only necessary for RPM builds. By default, '`configure`' will search for this tool.

RPMBUILD
Build RPM command, `rpmbuild(1)`. This is only necessary for RPM builds. By default, '`configure`' will search for this tool. `rpm(1)` will be used instead of `rpmbuild(1)` only if `rpmbuild(1)` cannot be found.

DPKG Dpkg comand, `dpkg(1)`. This command is used for building Debian packages. By default, '`configure`' will search for this tool.

DPKG_SOURCE
Dpkg-source command, `dpkg-source(1)`. This command is used for building Debian dsc packages. By default, '`configure`' will search for this tool.

DPKG_BUILDPACKAGE
Dpkg-buildpackage command, `dpkg-buildpackage(1)`. This command is used for building Debian deb packages. By default, '`configure`' will search for this tool.

DEB_BUILD_ARCH
Debian build architecture. This variable is used for building Debian packages. The default is the autoconf build architecture.

DEB_BUILD_GNU_CPU
Debian build cpu. This variable is used for building Debian packages. The default is the autoconf build cpu.

DEB_BUILD_GNU_SYSTEM
Debian build os. This variable is used for building Debian packages. The default is the autoconf build os.

DEB_BUILD_GNU_TYPE
: Debian build alias. This variable is used for building Debian packages. The default is the autoconf build alias.

DEB_HOST_ARCH
: Debian host architecture. This variable is used for building Debian packages. The default is the autoconf host architecture.

DEB_HOST_GNU_CPU
: Debian host cpu. This variable is used for building Debian packages. The default is the autoconf host cpu.

DEB_HOST_GNU_SYSTEM
: Debian host os. This variable is used for building Debian packages. The default is the autoconf host os.

DEB_HOST_GNU_TYPE
: Debian host alias. This variable is used for building Debian packages. The default is the autoconf host alias.

LDCONFIG
: Configure loader command, `ldconfig(8)`. Command used to configure the loader when libraries are installed. By default, '`configure`' will search for this tool.

DESTDIR
: Cross build root directory. Specifies the root directory for build and installation.

DEPMOD
: Build kernel module dependencies command, `depmod(8)`. This is used during installation of kernel modules to a running kernel to rebuild the modules dependency database. By default, '`configure`' will search for this tool.

MODPROBE
: Probe kernel module dependencies command, `modprobe(8)`. This is used during installation of kernel modules to a running kernel to remove old modules. By default, '`configure`' will search for this tool.

LSMOD
: List kernel modules command, `lsmod(8)`. This is used during installation of kernel modules to a running kernel to detect old modules for removal. By default, '`configure`' will search for this tool.

LSOF
: List open files command, `lsof(1)`. This is used during installation of kernel modules to a running kernel to detect old modules for removal. Processes owning the old kernel modules will be killed and the module removed. If the process restarts, the new module will be demand loaded. By default, '`configure`' will search for this tool.

GENKSYMS
: Generate kernel symbols command, `genksyms(8)`. This is used for generating module symbol versions during build. By default, '`configure`' will search for this tool.

KGENKSYMS
: Linux 2.6 generate kernel symbols command, `genksyms(8)`. This is used for generating module symbol version during build. By default, '`configure`' will search for this tool.

OBJDUMP
: Object dumping command, `objdump(1)`. This is used for listing information about object files. By default, '`configure`' will search for this tool.

NM
: Object symbol listing command, `nm(1)`. This is used for listing information about object files. By default, '`configure`' will search for this tool.

MODPOST_CACHE
: Cache file for `modpost(1)`. The version of the `modpost.sh` script that ships with each package can cache information to a cache file to speed multiple builds. This environment variable is used to specify a cache file.

AUTOM4TE
: Autom4te command, `autom4te(1)`. This is the executable used by `autotest` for pre- and post-installation checks. By default, '`configure`' will search for this tool.

AUTOTEST
: Autotest macro build command, `autom4te(1)`. This is the executable used by `autotest` for pre- and post-installation checks. By default, '`configure`' will search for this tool.

8.3.5.3 Build

To build from the tar ball, See Section 8.4.3 [Building from the Tar Ball], page 117.

8.4 Building

8.4.1 Building from the Source RPM

If you have downloaded the necessary source RPM (see Section 8.2.5 [Downloading the Source RPM], page 92), then the following instructions will rebuild the binary RPMs on your system. Once the binary RPMs are rebuilt, you may install them as described above (see Section 8.5.1 [Installing the Binary RPM], page 118).

The source RPM is rebuilt to binary RPMs as follows:

```
% wget http://www.openss7.org/rpms/SRPMS/streams-0.9.2.4-1.src.rpm
% rpmbuild --rebuild -vv streams-0.9.2.4-1.src.rpm
```

The rebuild process can also recognize a number of options that can be used to tweak the resulting binaries, See Section 8.3.3 [Configuring the Source RPM], page 98. These options are provided on the `rpm(1)` command line. For example:

```
% rpmbuild --rebuild -vv --target athlon-redhat-linux \
  --define "_kversion 2.4.20-28.7" \
  -- streams-0.9.2.4-1.src.rpm
```

will rebuild binary RPM for the '`2.4.20-28.7`' kernel for the '`athlon`' architecture.[19]

Installation

To install the resulting binary RPM, See Section 8.5.1 [Installing the Binary RPM], page 118.

8.4.2 Building from the Debian DSC

If you have downloaded the necessary Debian DSC (see Section 8.2.6 [Downloading the Debian DSC], page 93), then the following instructions will rebuild the binary DEBs on your system. Once the binary DEBs are rebuilt, you may install them as described above (see Section 8.5.2 [Installing the Debian DEB], page 119).

The Debian DSC is rebuilt to binary DEBs as follows:

```
% wget http://www.openss7.org/debian/streams_0.9.2.4-0.dsc
% wget http://www.openss7.org/debian/streams_0.9.2.4-0.tar.gz
% dpkg-buildpackage -v streams_0.9.2.4-0.dsc
```

The rebuild process can also recognize a number of options that can be used to tweak the resulting binaries, See Section 8.3.4 [Configuring the Debian DSC], page 104. These options are provided in the environment variable *BUILD_DPKGOPTIONS* and have the same form as the options to '`configure`', See Section 8.3.5 [Configuring the Tar Ball], page 104. For example:

```
% BUILD_DEBOPTIONS='
        --with-k-release=2.4.20-28.7
        --host=athlon-debian-linux-gnu'
  dpkg-buildpackage -v \
  streams_0.9.2.4-0.dsc
```

will rebuild binary DEB for the '`2.4.20-28.7`' kernel for the '`athlon`' architecture.[20]

Installation

To install the resulting binary DEB, See Section 8.5.2 [Installing the Debian DEB], page 119.

8.4.3 Building from the Tar Ball

If you have downloaded the tar ball (see Section 8.2.7 [Downloading the Tar Ball], page 93), then the following instructions will rebuild the package on your system. (Note that the build process does not required `root` privilege.)

[19] Note that the '`_kversion`' of '`2.4.20-28.7`' is only an example.

[20] Note that the '`_kversion`' of '`2.4.20-28.7`' is only an example.

8.4.3.1 Native Build

Following is an example of a native build against the running kernel:

```
% wget http://www.openss7.org/streams-0.9.2.4.tar.bz2
% tar -xjvf streams-0.9.2.4.tar.bz2
% pushd streams-0.9.2.4
% ./configure
% make
% popd
```

8.4.3.2 Cross-Build

Following is an example for a cross-build. The kernel release version must always be specified for a cross-build.[21] If you are cross-building, specify the root for the build with environment variable *DESTDIR*. The cross-compile host must also be specified if different from the build host. Either the compiler and other tools must be in the usual places where GNU `autoconf(1)` can find them, or they must be specified with declarations such as '`CC=/usr/lib/ppc-linux/gcc`' on the '`configure`' command line.

```
% wget http://www.openss7.org/streams-0.9.2.4.tar.bz2
% tar -xjvf streams-0.9.2.4.tar.bz2
% pushd streams-0.9.2.4
% ./configure DESTDIR="/some/other/root" \
--with-k-release=2.4.18 --host sparc-linux
% make
% popd
```

8.5 Installing

8.5.1 Installing the Binary RPM

If you have downloaded the necessary binary RPMs (see Section 8.2.3 [Downloading the Binary RPM], page 89), or have rebuilt binary RPMs using the source RPM (see Section 8.4.1 [Building from the Source RPM], page 116), then the following instructions will install the RPMs on your system. For additional information on `rpm(1)`, see `rpm(8)`.

```
% pushd RPMS/i686
% rpm -ihv streams-*-0.9.2.4-1.7.2.i686.rpm
```

You must have the correct binary RPMs downloaded or built for this to be successful.

Some of the packages are relocatable and can have final installation directories altered with the '`--relocate`' option to `rpm(1)`, see `rpm(8)`. For example, the following will relocate the documentation and info directories:

[21] Because it *is* a cross-build, the kernel version on the build machine is unlikely to be the kernel version of the target machine, except by coincidence.

```
% pushd RPMS/i686
% rpm -ihv \
        --relocate '/usr/share/doc=/usr/local/share/doc' \
        --relocate '/usr/share/info=/usr/local/share/info' \
        -- streams-doc-0.9.2.4-1.7.2.i686.rpm
```

The previous example will install the '`streams-doc`' package by will relocate the documentation an info directory contents to the '`/usr/local`' version.

8.5.2 Installing the Debian DEB

If you have downloaded the necessary Debian DEBs (see Section 8.2.4 [Downloading the Debian DEB], page 91), or have rebuild binary DEBs using the Debian DSC (see Section 8.4.2 [Building from the Debian DSC], page 117), then the following instructions will install the DEBs on your system. For additional information see `dpkg(8)`.

```
% pushd debian
% dpkg -iv streams-*_0.9.2.4-0_*.deb
```

You must have the correct '`.deb`' files downloaded or build for this to be successful.

8.5.3 Installing the Tar Ball

After the build process (see Section 8.4.3 [Building from the Tar Ball], page 117), installation only requires execution of one of two `automake(1)` targets:

'`make install`'
: The '`install`' `automake(1)` target will install all the components of the package. Root privilege is required to successfully invoke this target.

'`make install-strip`'
: The '`install-strip`' `automake(1)` target will install all the components of the package, but will strip unnecessary information out of the objects and compress manual pages. Root privilege is required to successfully invoke this target.

8.6 Removing

8.6.1 Removing the Binary RPM

To remove an installed version of the binary RPMs (whether obtained from the OpenSS7 binary RPM releases, or whether created by the source RPM), execute the following command:

```
% rpm -evv `rpm -qa | grep '^streams-'`
```

For more information see `rpm(1)`.

8.6.2 Removing the Debian DEB

To remove and installed version of the Debian DEB (whether obtained from the OpenSS7 binary DEB releases, or whether created by the Debian DSC), execute the following command:

```
% dpkg -ev `dpkg -l | grep '^streams-'`
```

For more information see dpkg(8).

8.6.3 Removing the Source RPM

To remove all the installed binary RPM build from the source RPM, see Section 8.6.1 [Removing the Binary RPM], page 119. Then simply remove the binary RPM package files and source RPM file. A command such as:

```
% find / -name 'streams-*.rpm' -type f -print0 | xargs --null rm -f
```

should remove all 'streams' RPMs from your system.

8.6.4 Removing the Debian DSC

To remove all the installed binary DEB build from the Debian DSC, see Section 8.6.2 [Removing the Debian DEB], page 119. Then simply remove the binary DEB package files and Debian DSC file. A command such as:

```
% find / \( -name 'streams-*.deb' \
         -o -name 'streams-*.dsc' \
         -o -name 'streams-*.tar.* \
         \) -type f -print0 | xargs --null rm -f
```

should remove all 'streams' DEBs, DSCs and TARs from your system.

8.6.5 Removing the Tar Ball

To remove a version installed from tar ball, change to the build directory where the package was built and use the 'uninstall' automake(1) target as follows:

```
% cd /usr/src/streams
% make uninstall
% cd ..
% rm -fr streams-0.9.2.4
% rm -f streams-0.9.2.4.tar.gz
% rm -f streams-0.9.2.4.tar.bz2
```

If you have inadvertently removed the build directory and, therefore, no longer have a configured directory from which to execute 'make uninstall', then perform all of the steps for configuration and installation (see Section 8.5.3 [Installing the Tar Ball], page 119) except the final installation and then perform the steps above.

8.7 Loading

8.7.1 Normal Module Loading

When Linux Fast-STREAMS installs, modules and drivers belonging to release packages are normally configured for demand loading. The 'install' and 'install-strip' automake(1)

targets will make the necessary changes to the '`/etc/modules.conf`' file and place the modules in an appropriate place in '`/lib/modules/2.4.20-28.7/streams`'. The '`make install`' process should have copied the kernel module files '`streams-*.o`' to the directory '`/lib/modules/2.4.20-28.7/streams`'. This means that to load any of these modules, you can simply execute, for example, '`modprobe stream-`*`somedriver`*'.[22]

8.7.1.1 Linux Fast-STREAMS Module Loading

The '`streams`' demand load system supports both the old `kerneld` and the new `kmod` mechanisms for demand loading kernel modules.

The convention for '`streams`' kernel loadable object files is:

- Their name start with "streams-".
- They are placed in '`/lib/modules/2.4.20-28.7/streams/`', where '`2.4.20-28.7`' is an example kernel version.

If your kernel has been built using the '`kerneld`' daemon, then '`streams`' kernel modules will automatically load as soon as the *STREAMS* module is pushed or the driver is opened. The '`make install`' process makes the necessary changes to the '`/etc/modules.conf`' file. After the install, you will see lines like the following added to your '`/etc/modules.conf`' file:

```
prune modules.streams
if -f /lib/modules/‘uname -r‘/modules.streams
include /lib/modules/‘uname -r‘/modules.streams
endif
```

which will provide for demand loading of the modules if they have been built and installed for the running kernel. The '`/lib/modules/‘uname -r‘/modules.streams`' file looks like this:

```
alias char-major-245  streams-some_driver
alias char-major-246  streams-other_driver
```

Note that *STREAMS* modules are not listed in this file, but will be loaded by name using '`kerneld`' if available.

Linux Fast-STREAMS has a wider range of kernel module loading mechanisms than is provided by the deprecated *LiS*. For mechanisms used for kernel module loading under *Linux Fast-STREAMS*, See Section "Top" in *Linux Fast-STREAMS Reference Manual*.

8.7.1.2 Linux STREAMS Module Loading

LiS is deprecated and this section has been deleted.

8.8 Maintenance

[22] Note that the '`_kversion`' of '`2.4.20-28.7`' is only an example.

8.8.1 Makefile Targets

automake(1) has many targets, not all of which are obvious to the casual user. In addition, *OpenSS7* automake(1) files have additional rules added to make maintaining and releasing a package somewhat easier. This list of targets provides some help with what targets can be invoked, what they do, and what they hope to achieve. The available targets are as follows:

8.8.1.1 User Targets

The following are normal targets intended to be invoked by installers of the package. They are concerned with compiling, checking the compile, installing, checking the installation, and removing the package.

'[all]' This is also the default target. It compiles the package and all release packages selected by 'configure'. This is performed after configuring the source with 'configure'. A 'Makefile' stub is provided so that if the package has not had autoreconf(1) run (such as when checked out from CVS, the package will attempt to run 'autoreconf -fiv'.

All *OpenSS7 Project* packages are configured without maintainer mode and without dependency tracking by default. This speeds compilation of the package for one-time builds. This also means that if you are developing using the source package (edit-compile-test cycle), changes made to source files will not cause the automatic rebuilding due to dependencies. There are two ways to enable dependency tracking: specify '--enable-maintainer-mode' to 'configure'; or, specify '--enable-dependency-tracking' to 'configure'. I use the former during my edit-compile-test cycle.

This is a standard *GNU* automake(1) makefile target. This target does not require root privilege.

'check' All *OpenSS7 Project* release packages provide check scripts for the check target. This step is performed after compiling the package and will run all of the 'check' programs against the compiled binaries. Which checks are performed depends on whether '--enable-maintainer-mode' was specified to 'configure'. If in maintainer mode, checks that assist with the release of the package will be run (such as checking that all manual pages load properly and that they have required sections.) We recommend running the check stage before installing, because it catches problems that might keep the installed package from functioning properly.

Another way to enable the greater set of checks, without invoking maintainer mode, is to specify '--enable-checks' to 'configure'. For more information, see Section 9.1.1 [Pre-installation Checks], page 133.

This is a standard *GNU* automake(1) makefile target, although the functions performed are customized for the *OpenSS7 Project*. This target does not require root privilege.

'install'
'install-strip'
The 'install' target installs the package by installing each release package. This target also performs some actions similar to the pre- and post-install scripts

used by packaging tools such as `rpm(1)` or `dpkg(1)`. The '`install-strip`' target strips unnecessary symbols from executables and kernel modules before installing.

This is a standard *GNU* `automake(1)` makefile target. This target requires root privilege.

'`installcheck`'
: All *OpenSS7 Project* packages provide test scripts for the '`installcheck`' target. Test scripts are created and run using `autotest` (part of the `autoconf(1)` package). Which test suites are run and how extensive they are depends on whether '`--enable-maintainer-mode`' was specified to '`configure`'. When in maintainer mode, all test suites will be run. When not in maintainer mode, only a few post-install checks will be performed, but the test suites themselves will be installed in '`/usr/libexec/streams`'[23] for later use.

  This is a standard *GNU* `automake(1)` makefile target. This target might require root privilege. Tests requiring root privilege will be skipped when run as a regular user. Tests requiring regular account privileges will be skipped when run as root.

'`retest`'
: To complement the '`installcheck`' target above, all *OpenSS7 Project* packages provide the '`retest`' target as a means to rerun failed conformance test suite test cases. The '`retest`' target is provided because some test cases in the test suites have delicate timing considerations that allow them to fail sporadically. Invoking this target will retest the failed cases until no cases that are not expected failures remain.

  This is an *OpenSS7 Project* specific makefile target. As with '`installcheck`', this target might require root privilege. Tests requiring root privilege will be skipped when run as a regular user. Tests requiring regular account privileges will be skipped when run as root.

'`uninstall`'
: This target will reverse the steps taken to install the package. This target also performs pre- and post- erase scripts used by packaging tools such as *rpm* or *dpkg*. You need to have a configured build directory from which to execute this target, however, you do not need to have compiled any of the files in that build directory.[24]

  The '`uninstall`' target unfortunately removes add-on packages in the same order in which they were installed. This is not good for the *OpenSS7 Master Package*, where the '`remove`' target should be used instead.

  This is a standard *GNU* `automake(1)` makefile target. This target requires root privilege.

'`remove`'
: This target is like '`uninstall`' with the exception that it removes add-on packages in the reverse order that installation was performed.[25]

---

[23] '`/usr/libexec/streams`' is just an example, the actual location is '`${libexecdir}/${PACKAGE}`', which varies from distribution to distribution (as some distributions such as Mandriva do not have a libexec directory).

[24] Therefore, it is possible to download the package, configure it, and then uninstall it. This is handy if you do not have the sources used to build and install the package immediately available.

[25] This is useful from the *OpenSS7 Master Package*.

This is an *OpenSS7 Project* specific makefile target. This target requires root privilege.

8.8.1.2 Maintainer Targets

The following targets are targets intended for use by maintainers of the package, or those responsible for release and packaging of a derivative work of the package. Some of these targets are only effective when maintainer mode has been invoked ('`--enable-maintainer-mode`' specified to '`configure`'.)

'`dist`' Creates a distribution package (tarball) in the top level build directory. *OpenSS7 Project* packages distribute two archives: a '`gzip tar`' archive and a '`bzip tar`' archive. These archives will have the name '`streams-0.9.2.4.tar.gz`' and '`streams-0.9.2.4.tar.bz2`'.

This is a standard *GNU* `automake(1)` makefile target. This target does not require root privilege.

'`distcheck`'
This target is intended for use when releasing the package. It creates the `tar(1)` archives above and then unpacks the tarball in a source directory, configures in a separate build directory, compiles the package, installs the package in a separate install directory, tests the install package to ensure that some components work, and, finally, uses the unpacked source tree to build another tarball. If you have added or removed files from the package, this is a good way to ensure that everything is still stable for release.

This is a standard *GNU* `automake(1)` makefile target. This target does not require root privilege.

8.8.1.3 Clean Targets

'`mostlyclean`'
Cleans out most of the files from the compile stage. This target is helpful if you have not enabled dependency tracking and need to recompile with changes.

This is a standard *GNU* `automake(1)` makefile target. This target does not require root privilege.

'`clean`' Cleans all the files from the build directory generated during the '`make [all]`' phase. It does not, however, remove files from the directory left there from the '`configure`' run. Use the '`distclean`' target to remove those too.

This is a standard *GNU* `automake(1)` makefile target. This target might require root privilege if the '`installcheck`' target or the `testsuite` was invoked with root privilege (leaving files belonging to root).

'`distclean`'
This target cleans out the directories left behind by '`distcheck`' and removes all the '`configure`' and generated files from the build directory. This will effectively remove all the files in the build directory, with the except of files that belong to you or some other process.

This is a standard *GNU* `automake(1)` makefile target. This target might require root privilege if the '`installcheck`' target or the `testsuite` was invoked with root privilege (leaving files belonging to root).

'`maintainer-clean`'
: This target not only removes files from the build directory, it removes generated files from the source directory as well. Care should be taken when invoking this target, because it removes files generated by the maintainer and distributed with the archive that might require special tools to regenerate. These special tools might only be available to the maintainer.[26] It also means that you probably need a full blown Linux system to rebuild the package. For more information, see Section 8.2.8 [Downloading from CVS], page 94.

  This is a standard *GNU* `automake(1)` makefile target. This target might require root privilege if the '`installcheck`' target or the `testsuite` was invoked with root privilege (leaving files belonging to root).

'`check-clean`'
: This target removes log files left behind by the '`check`' target. By default, the check scripts append to log files in the top level build directory. This target can be used to clean out those log files before the next run.

  This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

8.8.1.4 Manual Page Targets

The following targets are used to build, install and uninstall just the manual pages from the distribution. These targets are good for creating a distribution of just the manual pages. When building atop multiple packages, these targets recurse down through each package.

'`mans`'
: Build all of the manual pages. This involves performing parameter substitution on manual pages and optionally cooking the manual pages if '`--with-cooked-manpages`' was requested during configuration.

'`install-mans`'
: Installs the manual pages under *DESTDIR*. Specify *DESTDIR* to place the manual pages wherever you see fit. If *DESTDIR* is not specified on the command line, the manual pages will be installed in the normal installation directory.

'`uninstall-mans`'
: Uninstalls the manual pages from *DESTDIR*. Specify *DESTDIR* to indicate where to remove the manual pages from. If *DESTDIR* is not specified on the command line, the manual pages will be removed from the normal installation directory.

8.8.1.5 Release Targets

The following are targets used to generate complete releases into the package distribution directory. These are good for unattended and NFS builds, which is what I use them for.

[26] Theoretically this is true, however, the *OpenSS7 Project* does not use any maintainer programs that are not generally available (i.e. open source).

Also, when building from atop multiple packages, these targets also recurse down through each package.

'`release`' Build all of the things necessary to generate a release. On an `rpm(1)` system this is the distribution archives, the source rpm, and the architecture dependent and architecture independent binary rpms. All items are placed in the package distribution directory that can be specified with the '`--with-pkg-distdir=DIR`' option to '`configure`'.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

'`forced-release`'
The '`release`' target will not regenerate any files that already exist in the package distribution directory. This forced target will.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

'`release-sign`'
You will be prompted for a password, unless to specify it to make with the *GNUPGPASS* variable. For unattended or non-interactive builds with signing, you can do that as: '`make GNUPGPASS=mypasswd release-sign`'

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

'`forced-release-sign`'
The '`release-sign`' target will not regenerate any files that already exist in the package distribution directory. This forced target will.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

'`release-clean`'
This target will remove all distribution files for the current package from the package distribution directory.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

8.8.1.6 Logging Targets

For convenience, to log the output of a number of targets to a file, log targets are defined. The log file itself is used as the target to make, but make invokes the target minus a '`.log`' suffix. So, for example, to log the results of target '`foo`', invoke the target '`foo.log`'. The only target that this does not apply to is '`compile.log`'. When you invoke the target '`compile.log`' a simple `automake(1)` is invoked and logged to the file '`compile.log`'. The '`foo.log`' rule applies to all other targets. This does not work for all targets, just a selected few.[27] Following are the logging targets:

[27] Note that because logging targets invoke a pipe, `automake(1)` does not return the correct return status (always returns success if the `tee(1)` operation is successful). Therefore, these targets should not be invoked by scripts that need to use the return value from `automake(1)`.

Common Logging Targets

Common logging targets correspond to normal user `automake(1)` makefile targets as follows:

'`compile.log`'
: This is an *OpenSS7 Project* specific makefile target, but it invokes the standard *GNU* `automake(1)` makefile target '`[all]`'.

'`check.log`'
: This is an *OpenSS7 Project* specific makefile target, but it invokes the standard *GNU* `automake(1)` makefile target '`check`'.

'`install.log`'
: This is an *OpenSS7 Project* specific makefile target, but it invokes the standard *GNU* `automake(1)` makefile target '`install`'.

'`installcheck.log`'
: This is an *OpenSS7 Project* specific makefile target, but it invokes the standard *GNU* `automake(1)` makefile target '`installcheck`'.

'`uninstall.log`'
: This is an *OpenSS7 Project* specific makefile target, but it invokes the standard *GNU* `automake(1)` makefile target '`uninstall`'.

'`remove.log`'
: This is an *OpenSS7 Project* specific makefile target, that invokes the *OpenSS7 Project* '`remove`' target.

Maintainer Logging Targets

Maintainer logging targets correspond to maintainer mode `automake(1)` makefile targets as follows:

'`dist.log`'
: This is an *OpenSS7 Project* specific makefile target, but it invokes the standard *GNU* `automake(1)` makefile target '`dist`'.

'`distcheck.log`'
: This is an *OpenSS7 Project* specific makefile target, but it invokes the standard *GNU* `automake(1)` makefile target '`distcheck`'.

'`srpm.log`'
: This is an *OpenSS7 Project* specific makefile target, that invokes the *OpenSS7 Project* '`srpm`' target.

'`rebuild.log`'
: This is an *OpenSS7 Project* specific makefile target, that invokes the *OpenSS7 Project* '`rebuild`' target.

'`resign.log`'
: This is an *OpenSS7 Project* specific makefile target, that invokes the *OpenSS7 Project* '`resign`' target.

'`release.log`'
: This is an *OpenSS7 Project* specific makefile target, that invokes the *OpenSS7 Project* '`release`' target.

‘release-sign.log’
: This is an *OpenSS7 Project* specific makefile target, that invokes the *OpenSS7 Project* ‘release-sign’ target.

If you want to add one, simply add it to *LOGGING_TARGETS* in ‘Makefile.am’.

8.8.1.7 Problem Report Targets

To ease problem report generation, all logging targets will automatically generate a problem report suitable for mailing in the file ‘*target*.pr’ for target ‘*target*.log’. This problem report file is in the form of an email and can be sent using the included send-pr script or by invoking the ‘send-pr’ makefile target.

There are two additional problem report targets:

‘pr’
: The ‘pr’ target is for independently generating a problem report outside of the build or installation process. The target will automatically generate a problem report skeleton suitable for editing and mailing in the file ‘problem.pr’. This problem report file is in the form of an email and can be edited and sent directly, or sent using the included send-pr script or by invoking the ‘send-pr’ target.

  This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

‘send-pr’
: The ‘send-pr’ target is for finalizing and mailing a problem report generated either inside or outside the build and installation process. The target will automatically finalize and mail the ‘problem.pr’ problem report if it has changed since the last time that ‘send-pr’ was invoked.

  This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege (unless the problem report file was generated as root).

8.8.1.8 Release Archive Targets

The following targets are used to generate and clean distribution archive and signature files. Whereas the ‘dist’ target affects archives in the top build directory, the ‘release-archive’ targets affects archives in the package distribution directory (either the top build directory or that specified with ‘--with-pkg-distdir=DIR’ to ‘configure’).

You can change the directory to which packages are distributed by using the ‘--with-pkg-distdir=DIR’ option to ‘configure’. The default directory is the top build directory.

‘release-archives’
: This target creates the distribution archive files if they have not already been created. This not only runs the ‘dist’ target, but also copies the files to the distribution directory, which, by default is the top build directory.

  The files generated are named:

  ‘streams-0.9.2.4.tar.gz’ and ‘streams-0.9.2.4.tar.bz2’

  You can change this distribution directory with the ‘--with-pkg-distdir’ option to ‘configure’. See ‘./configure --help’ for more details on options.

  This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

‘`release-sign-archives`’
: This target is like ‘`release-archives`’, except that it also signs the archives using a *GPG* detached signature. You will be prompted for a password unless you pass the *GNUPGPASS* variable to make. For automated or unattended builds, pass the *GNUPGPASS* variable like so:

  ‘`make GNUPGPASS=mypasswd release-sign-archives`’

  Signature files will be named:

  ‘`streams-0.9.2.4.tar.gz.asc`’ and ‘`streams-0.9.2.4.tar.bz2.asc`’

  These files will be moved to the package distribution directory with the plain text archives.

  This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

‘`release-clean-archives`’
: This target will clean the release archives and signature files from the package distribution directory.

  This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

8.8.1.9 RPM Build Targets

On `rpm(1)` systems, or systems sporting rpm packaging tools, the following targets are used to generate `rpm(1)` release packages. The epoch and release number can be controlled by the contents of the ‘`.rpmepoch`’ and ‘`.rpmrelease`’ files, or with the ‘`--with-rpm-epoch=EPOCH`’ and ‘`--with-rpm-release=RELEASE`’ options to ‘`configure`’. See ‘`configure --help`’ for more information on options. We always use release number ‘`1`’. You can use release numbers above ‘`1`’.

‘`srpm`’
: This target generates the source rpm for the package (without signing the source rpm). The source rpm will be named: ‘`streams-0.9.2.4-1.srpm`’.

  This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

‘`rpms`’
: This target is responsible for generating all of the package binary rpms for the architecture. The binary rpms will be named:

  ‘`streams-*-0.9.2.4-1.*.rpm`’

  where the stars indicate the subpackage and the architecture. Both the architecture specific subpackages (binary objects) and the architecture independent (‘`.noarch`’) subpackages will be built unless the the former was disabled with the option ‘`--disable-arch`’, or the later with the option ‘`--disable-indep`’, passed to ‘`configure`’.

  This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

‘`sign`’
‘`srpm-sign`’
: These two targets are the same. When invoked, they will add a signature to the source rpm file, provided that the file does not already have a signature.

You will be prompted for a password if a signature is required. Automated or unattended builds can be achieved by using the `emake` expect script, included in '`${srcdir}/scripts/emake`'.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

'`rebuild`' This target accepts searches out a list of kernel names from the '`${DESTDIR}/lib/modules`' directory and builds rpms for those kernels and for each of a set of architectures given in the *AM_RPMTARGETS* variable to make. This is convenience target for building a group of rpms on a given build machine.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

'`resign`' This target will search out and sign, with a *GPG* signature, the source rpm, and all of the binary rpms for this package that can be found in the package distribution directory. This target will prompt for a *GPG* password. Automated or unattended builds can be achieved with the `emake` expect script located here: '`${srcdir}/scripts/emake`'.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

8.8.1.10 Debian Build Targets

On Debian systems, or systems sporting Debian packaging tools, the following targets are used to generate Debian release packages. The release number can be controlled by the contents of the '`.debrelease`' file, or with the '`--with-debrelease=RELEASENUMBER`' option to '`configure`'. See '`configure --help`' for more information on options.

'`dsc`' This target will build the Debian source change package ('`.dsc`' file). We use release number '`0`' so that the entire tarball is included in the '`dsc`' file. You can use release number '`1`' for the same purposes. Release numbers above '`1`' will not include the entire tarball. The '`.dsc`' file will be named: '`streams_0.9.2.4-0.dsc`'.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

'`sigs`' This target signs the '`.deb`' files. You will be prompted for a password, unless to specify it to make with the *GNUPGPASS* variable.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

'`debs`' This target will build the Debian binary package ('`.deb`' file) from the '`.dsc`' created above. (This target will also create the '`.dsc`' if it has not been created already.) The subpackage '`.deb`' files will be named: '`streams-*_0.9.2.4-0_*.deb`', where the stars indicate the subpackage and the architecture.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

‘`csig`’ This target signs the ‘`.dsc`’ file. You will be prompted for a password, unless to specify it to make with the *GNUPGPASS* variable.

This is an *OpenSS7 Project* specific makefile target. This target does not require root privilege.

8.8.1.11 Documentation Targets

On systems that have `doxygen(1)` documentation tool, the following targets are used to generate doxygen html documentation:

‘`doxy`’ This target generates `doxygen(1)` documetnation from suitably marked sources. File containing the necessary documentation marks are discovered automatically by `configure`. Doxygen documentation can be generated bus is not distributed. Documentation is cerated in the subdirectory ‘`doc/html`’.

9 Troubleshooting

9.1 Test Suites

9.1.1 Pre-installation Checks

Most *OpenSS7* packages, including the *Linux Fast-STREAMS* package, ship with pre-installation checks integral to the build system. Pre-installation checks include check scripts that are shipped in the '`scripts`' subdirectory as well as specialized `make` targets that perform the checks.

When building and installing the package from *RPM* or *DEB* source packages (see Section 8.4.1 [Building from the Source RPM], page 116; and Section 8.4.2 [Building from the Debian DSC], page 117), a fundamental set of post-compile, pre-installation checks are performed prior to building binary packages. This is performed automatically and does not require any special actions on the part of the user creating binary packages from source packages.

When building and installing the package from *tarball* (see Section 8.4.3 [Building from the Tar Ball], page 117; and Section 8.5.3 [Installing the Tar Ball], page 119), however, pre-installation checks are only performed if specifically invoked by the builder of the package. Pre-installation checks are invoked after building the package and before installing the package. Pre-installation checks are performed by invoking the '`check`' or '`check.log`' target to `make` when building the package, as shown in Example 9.1.

```
% wget http://www.openss7.org/streams-0.9.2.4.tar.bz2
% tar -xjvf streams-0.9.2.4.tar.bz2
% pushd streams-0.9.2.4
% ./configure
% make
% make check  # <------- invoke pre-installation checks
% popd
```

Example 9.1: *Invoking Pre-Installation Checks*

Pre-installation checks fall into two categories: *System Checks* and *Maintenance Checks*.

9.1.1.1 Pre-Installation System Checks

System Checks are post-compilation checks that can be performed before installing the package that check to ensure that the compiled objects function and will be successfully installed. When the '`--enable-maintainer-mode`' option has not been passed to `configure`, only *System Checks* will be performed.

For example, the steps shown in Example 9.2 will perform *System* checks.

```
% wget http://www.openss7.org/streams-0.9.2.4.tar.bz2
% tar -xjvf streams-0.9.2.4.tar.bz2
% pushd streams-0.9.2.4
% ./configure
% make
% make check  # <------ invokes System pre-installation checks
% popd
```

Example 9.2: *Invoking System Checks*

9.1.1.2 Pre-Installation Maintenance Checks

Maintenance Checks include all *System Checks*, but also checks to ensure that the kernel modules, applications programs, header files, development tools, test programs, documentation, and manual pages conform to *OpenSS7* standards. When the '`--enable-maintainer-mode`' option has been passed to `configure`, *Maintenance Checks* will be performed.

For example, the steps shown in Example 9.3 will perform *Maintenance* checks.

```
% wget http://www.openss7.org/streams-0.9.2.4.tar.bz2
% tar -xjvf streams-0.9.2.4.tar.bz2
% pushd streams-0.9.2.4
% ./configure --enable-maintainer-mode
% make
% make check  # <------ invokes Maintenance pre-installation checks
% popd
```

Example 9.3: *Invoking Maintenance Checks*

9.1.1.3 Specific Pre-Installation Checks

A number of check scripts are provided in the '`scripts`' subdirectory of the distribution that perform both *System* and *Maintenance* checks. These are as follows:

`check_commands`

This check performs both *System* and *Maintenance* checks.

When performing *System* tests, the following tests are performed:

Unless cross-compiling, or unless a program is included in `AM_INSTALLCHECK_STD_OPTIONS_EXEMPT` every program in `bin_PROGRAMS`, `sbin_PROGRAMS`, and `libexec_PROGRAMS` is tested to ensure that the '`--help`', '`--version`', and '`--copying`' options are accepted. When cross-compiling is is not possible to execute cross-compiled binaries, and these checks are skipped in that case.

Script executables, on the other hand, can be executed on the build host, so, unless listed in `AM_INSTALLCHECK_STD_OPTIONS_EXEMPT`, every program in `dist_bit_SCRIPTS`, `dist_sbin_SCRIPTS`, and `pkglibexec_SCRIPTS` are tested to ensure that the '`--help`', '`--version`', and '`--copying`' options are accepted.

When performing *Maintenance* tests, `check_commands` also checks to ensure that a manual page exists in section 1 for every executable binary or script

that will be installed from `bin_PROGRAMS` and `dist_bin_SCRIPTS`. It also checks to ensure that a manual page exists in section 8 for every executable binary or script that will be installed from `sbin_PROGRAMS`, `dist_sbin_SCRIPTS`, `libexec_PROGRAMS`, and `pkglibexec_SCRIPTS`.

`check_decls`

This check only performs *Maintenance* checks.

It collects the results from the `check_libs`, `check_modules` and `check_headers` check scripts and tests to ensure every declaration of a function prototype or external variable contained in installed header files has a corresponding exported symbol from either a to be installed shared object library or a to be installed kernel module. Declarations are exempted from this requirement if their identifiers have been explicitly added to the `EXPOSED_SYMBOL` variable. If `WARN_EXCESS` is set to '`yes`', then the check script will only warn when excess declarations exist (without a corresponding exported symbol); otherwise, the check script will generate an error and the check will fail.

`check_headers`

This check only performs *Maintenance* checks.

When performing *Maintenance* tests, it identifies all of the declarations included in to be installed header files. It then checks to ensure that a manual page exists in sections 2, 3, 7 or 9, as appropriate, for the type of declaration. It also checks to see if a manual page source file exists in the source directory for a declaration that has not been included in the distribution. Function or prototype declarations that do not have a manual page in sections 2, 3, or 9 will cause the check to fail. Other declarations ('`variable`', '`externvar`', '`macro`', '`enumerate`', '`enum`', '`struct`', '`union`', '`typedef`', '`member`', etc.) will only warn if a manual page does not exist, but will not fail the check.

`check_libs`

This check only performs *Maintenance* checks.

When performing *Maintenance* tests, it checks that each exported symbol in each to be installed shared object library has a manual page in section 3. It also checks that each exported symbol has a '`function`', '`prototype`' or '`externvar`' declaration in the to be installed header files. A missing declaration or manual page will cause this check to fail.

`check_mans`

This check only performs *Maintenance* checks.

When performing *Maintenance* tests, it checks that to be install manual pages can be formatted for display without any errors or warnings from the build host `man` program. It also checks that required headings exist for manual pages according to the section in which the manual page will be installed. It warns if recommended headings are not included in the manual pages. Because some *RPM* distributions have manual pages that might conflict with the package manual pages, this check script also checks for conflicts with installed manual pages on the build host. This check script also checks to ensure that all to be

installed manual pages are used in some fashion, that is, they have a declaration, or exported symbol, or are the name of a kernel module or STREAMS module or driver, possibly capitalized.

Note that checking for conflicts with the build host should probably be included in the *System* checks (because *System* checks are performed before the source *RPM* `%install` scriptlet).

`check_modules`

This check performs both *System* and *Maintenance* checks.

When performing *System* tests, it checks each to be installed kernel module to ensure that all undefined symbols can be resolved to either the kernel or another module. It also checks whether an exported or externally declared symbol conflicts with an exported or externally declared symbol present in the kernel or another module.[1]

When performing *Maintenance* tests, this check script tests that each to be installed kernel module has a manual page in section 9 and that each exported symbol that does not begin with an underscore, and that belongs to an exported function or exported variable, has a manual page in section 9. It also checks to ensure that each exported symbol that does not begin with an underscore, and that belongs to an exported function or exported variable, has a '`function`', '`prototype`' or '`externvar`' declaration in the to be installed header files.

`check_streams`

This check performs only *Maintenance* checks.

When performing *Maintenance* tests, it checks that for each configured *STREAMS* module or driver, or device node, that a manual page exists in section 4 or section 7 as appropriate.

The output of the pre-installation tests are fairly self explanatory. Each check script saves some output to '*`name`*`.log`', where *name* is the name of the check script as listed above. A summary of the results of the test are display to standard output and can also be captured to the '`check.log`' file if the '`check.log`' target is used instead of the '`check`' target to `make`.

Because the check scripts proliferate '*`name`*`.log`' files throughout the build directory, a '`make check-clean`' `make` target has be provided to clean them out. '`make check-clean`' should be run before each successive run of '`make check`'.

9.1.2 Post-installation Checks

Most OpenSS7 packages ship with a compatibility and conformance test suite built using the '`autotest`' capabilities of '`autoconf`'. These test suites act as a wrapper for the compatibility and conformance test programs that are shipped with the package.

Unlike the pre-installation checks, the post-installation checks are always run complete. The only check that post-installation test scripts perform is to test whether they have been invoked with root privileges or not. When invoked as root, or as a plain user, some tests might be skipped that require root privileges, or that require plain user privileges, to complete successfully.

[1] This particular check has caught some name space pollution that has occurred in the 2.6.11 kernel.

9.1.2.1 Running Test Suites

There are several ways of invoking the conformance test suites:

1. The test suites can be run after installation of the package by invoking the '`make installcheck`' or '`make installcheck.log`' target. Some packages require that root privileges be acquired before invoking the package.
2. The test suites can be run from the distribution subdirectory after installation of the package by invoking the `testsuite` shell script directly.
3. The test suites can be run standalone from the '`libexec`' ('`/usr/libexec`') installation directory by invoking the `testsuite` shell script directly.

Typical steps for invoking the test suites directly from `make` are shown in Example 9.4.

```
% wget http://www.openss7.org/streams-0.9.2.4.tar.bz2
% tar -xjvf streams-0.9.2.4.tar.bz2
% pushd streams-0.9.2.4
% ./configure
% make
% make check  # <------ invokes System pre-installation checks
% make install
% sudo make installcheck # <------- invokes post-installation tests
% popd
```

Example 9.4: *Invoking System Checks*

When performing post-installation checks for the purposes of generating a problem report, the checks should always be performed from the build directory, either with '`make installcheck`' or by invoking `testsuite` directly from the '`tests`' subdirectory of the build directory. This ensures that all of the information known to `configure` and pertinent to the configuration of the system for which a test case failed, will be collected in the resulting '`testsuite.log`' file deposited upon test suite failure in the '`tests`' directory. This '`testsuite.log`' file can then be attached as part of the problem report and provides rich details to maintainers of the package. See also See Section 9.2 [Problem Reports], page 137, below.

Typical steps for invoking and installed `testsuite` standalone are shown in Example 9.5.

```
% [sudo] /usr/libexec/streams/testsuite
```

Example 9.5: *Invoking* `testsuite` *Directly*

When invoked directly, `testsuite` will generate a '`testsuite.log`' file in the current directory, and a '`testsuite.dir`' directory of failed tests cases and debugging scripts. For generating a problem report for failed test cases, see Section 9.2.4 [Stand Alone Problem Reports], page 140.

9.2 Problem Reports

9.2.1 Problem Report Guidelines

Problem reports in the following categories should include a log file as indicated in the table below:

‘`./configure`’
: A problem with the configuration process occurs that causes the ‘`./configure`’ command to fail. The problem report must include the ‘`config.log`’ file that was generated by `configure`.

‘`make compile.log`’
: A problem with the build process occurs that causes the ‘`make`’ command to fail. Perform ‘`make clean`’ and then ‘`make compile.log`’ and attach the ‘`config.log`’ and ‘`compile.log`’ files to the problem report.

‘`make check.log`’
: A problem occurs with the ‘`make check`’ target that causes it to fail. Perform ‘`make check-clean check.log`’ and attach the ‘`config.log`’, ‘`compile.log`’ and ‘`check.log`’ files to the problem report.

‘`sudo make install.log`’
: A problem occurs with ‘`sudo make install`’ that causes it to fail. Perform ‘`sudo make uninstall`’ and ‘`sudo make install.log`’ and attach the ‘`config.log`’, ‘`compile.log`’, ‘`check.log`’, and ‘`install.log`’ files to the problem report.

‘`[sudo] make installcheck.log`’
: A problem occurs with the ‘`make installcheck`’ target that causes the test suite to fail. Attach the resulting ‘`tests/testsuite.log`’ and ‘`installcheck.log`’ file to the problem report. There is no need to attach the other files as they are included in ‘`tests/testsuite.log`’.

‘`[sudo] make uninstall.log`’
: A problem occurs with the ‘`make uninstall`’ target that causes the test suite to fail. Perform ‘`sudo make uninstall.log`’ and attach the ‘`config.log`’, ‘`compile.log`’, ‘`check.log`’, ‘`install.log`’, ‘`installcheck.log`’, ‘`tests/testsuite.log`’ and ‘`uninstall.log`’ file to the problem report.

‘`[sudo] make remove.log`’
: A problem occurs with the ‘`make remove`’ target that causes the test suite to fail. Perform ‘`sudo make remove.log`’ and attach the ‘`config.log`’, ‘`compile.log`’, ‘`check.log`’, ‘`install.log`’, ‘`installcheck.log`’, ‘`tests/testsuite.log`’ and ‘`remove.log`’ file to the problem report.

For other problems that occur during the use of the *Linux Fast-STREAMS* package, please write a test case for the test suite that recreates the problem if one does not yet exist and provide a test program patch with the problem report. Also include whatever log files are generated by the kernel (`cmn_err(9)`) or by the `strerr(8)` or `strace(1)` facilities (`strlog(9)`).

9.2.2 Generating Problem Reports

The OpenSS7 Project uses the *GNU GNATS* system for problem reporting. Although the ‘`send-pr`’ tool from the *GNU GNATS* package can be used for bug reporting to the project’s

GNATS database using electronic mail, it is not always convenient to download and install the *GNATS* system to gain access to the '`send-pr`' tool.

Therefore, the *Linux Fast-STREAMS* package provides the '`send-pr`' shell script that can be used for problem reporting. The '`send-pr`' shell script can invoked directly and is a work-alike for the *GNU* '`send-pr`' tool.

The '`send-pr`' tool takes the same flags and can be used in the same fashion, however, whereas '`send-pr`' is an interactive tool[2], '`send-pr`' is also able to perform batch processing. Whereas '`send-pr`' takes its field information from local databases or from using the '`query-pr`' C-language program to query a remote database, the '`send-pr`' tool has the field database internal to the tool.

Problem reports can be generate using `make`, See Section 8.8.1.7 [Problem Report Targets], page 128. An example of how simple it is to generate a problem report is illustrated in Example 9.6.

```
% make pr
SEND-PR:
SEND-PR: send-pr:  send-pr was invoked to generate an external report.  An
SEND-PR: automated problem report has been created in the file named
SEND-PR: 'problem.pr' in the current directory.  This problem report can
SEND-PR: be sent to bugs@openss7.org by calling this script as
SEND-PR: '/home/brian/os7/scripts/send-pr --file="problem.pr"'.
SEND-PR:
SEND-PR: It is possible to edit some of the fields before sending on the
SEND-PR: problem report.  Please remember that there is NO WARRANTY.  See
SEND-PR: the file 'COPYING' in the top level directory.
SEND-PR:
SEND-PR: Please do not send confidential information to the bug report
SEND-PR: address.  Inspect the file 'problem.pr' for confidential
SEND-PR: information before mailing.
SEND-PR:
% vim problem.pr  # <--- follow instructions at head of file
% make send-pr
```

Example 9.6: *Invoking Problem Report Generation*

Using the '`make pr`' target to generate a problem report has the advantages that it will assemble any available '`*.log`' files in the build directory and attach them to the problem report.

9.2.3 Automatic Problem Reports

The *Linux Fast-STREAMS* package also provides a feature for automatic problem report generation that meets the problem report submission guidelines detailed in the preceding sections.

Whenever a logging makefile target (see Section 8.8.1.6 [Logging Targets], page 126) is invoked, if the primary target fails, the `send-pr` shell script is invoked to automatically

[2] '`send-pr`' launches the user's *EDITOR* to edit the problem report before submitting it.

generate a problem report file suitable for the corresponding target (as described above under see Section 9.2.1 [Problem Report Guidelines], page 138). An example is shown in Example 9.7.

```
% make compile.log
...
...
make[5]: *** [libXNSdrvs_a-ip.o] Error 1
make[5]: Leaving directory `/u6/buildel4/strxns'
make[4]: *** [all-recursive] Error 1
make[4]: Leaving directory `/u6/buildel4/strxns'
make[3]: *** [all] Error 2
make[3]: Leaving directory `/u6/buildel4/strxns'
make[2]: *** [all-recursive] Error 1
make[2]: Leaving directory `/u6/buildel4'
make[1]: *** [all] Error 2
make[1]: Leaving directory `/u6/buildel4'
SEND-PR:
SEND-PR: send-pr:  Make target compile.log failed in the compile stage.  An
SEND-PR: automated problem report has been created in the file named
SEND-PR: 'problem.pr' in the current directory.  This problem report can
SEND-PR: be sent to bugs@openss7.org by calling 'make send-pr'.
SEND-PR:
SEND-PR: It is possible to edit some of the fields before sending on the
SEND-PR: problem report.  Please remember that there is NO WARRANTY.  See
SEND-PR: the file 'COPYING' in the top level directory.
SEND-PR:
SEND-PR: Please do not send confidential information to the bug report
SEND-PR: address.  Inspect the file 'problem.pr' for confidential
SEND-PR: information before mailing.
SEND-PR:
% vim problem.pr  # <--- follow instructions at head of file
% make send-pr
```

Example 9.7: *Problem Report from Failed Logging Target*

9.2.4 Stand Alone Problem Reports

The *Linux Fast-STREAMS* package installs the `send-pr` script and its configuration file '`send-pr.config`' in '`${libexecdir}/streams`' along with the validation `testsuite`, see See Section 9.1 [Test Suites], page 133. As with the `testsuite`, this allows the `send-pr` script to be used for problem report generation on an installed system that does not have a build directory.

An example of invoking the package `testsuite` and then generating a problem report for failed cases is shown in Example 9.8.

```
% [sudo] /usr/libexec/streams/testsuite
% # test cases failed...
% /usr/libexec/streams/send-pr
SEND-PR:
SEND-PR: send-pr:  send-pr was invoked to generate an external report.  An
SEND-PR: automated problem report has been created in the file named
SEND-PR: 'problem.pr' in the current directory.  This problem report can
SEND-PR: be sent to bugs@openss7.org by calling this script as
SEND-PR: '/usr/libexec/streams/send-pr --file problem.pr'.
SEND-PR:
SEND-PR: It is possible to edit some of the fields before sending on the
SEND-PR: problem report.  Please remember that there is NO WARRANTY.  See
SEND-PR: the file 'COPYING' in the top level directory.
SEND-PR:
SEND-PR: Please do not send confidential information to the bug report
SEND-PR: address.  Inspect the file 'problem.pr' for confidential
SEND-PR: information before mailing.
SEND-PR:
% vim problem.pr  # <--- follow instructions at head of file
% /usr/libexec/streams/send-pr --file problem.pr
```

Example 9.8: *Invoking* `send-pr` *Directly*

The advantage of the approach shown in the example is that the `send-pr` script is capable of collecting the '`testsuite.log`' file and the failed test cases and debugging scripts from the '`testsuite.dir`' directory and including them in the problem report, as well as all package pertinent information from the installed '`send-pr.config`'.

9.3 Known Problems

The OpenSS7 Project does not ship software with known bugs. All bugs are unknown.

Verified behaviour is that behaviour that has been verified by conformance test suites that are shipped with the *Linux Fast-STREAMS* package.

Unverified behaviour may contain unknown bugs.

Please remember that there is **NO WARRANTY**.

See also Section 7.5 [Bugs], page 72, or file '`BUGS`' in the release directory.

Licenses

GNU Affero General Public License

The GNU Affero General Public License.
Version 3, 19 November 2007

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU Affero General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an

appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a. The work must carry prominent notices stating that you modified it, and giving a relevant date.
   b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
   c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
   d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

   a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
   b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange,

for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any

third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

   a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
   b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
   c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
   d. Limiting the use for publicity purposes of names of licensors or authors of the material; or
   e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
   f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license

to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

    Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar

in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as published by
the Free Software Foundation, either version 3 of the License, or (at
your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see http://www.gnu.org/licenses/.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see http://www.gnu.org/licenses/.

GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other

domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular

programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty

adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

   b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

   c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work,

but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with

the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or (at
your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see http://www.gnu.org/licenses/.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
program Copyright (C) year name of author
This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

GNU Lesser General Public License

GNU LESSER GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Terms and Conditions

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

   As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

   "The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

   An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

   A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

   The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

   The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

   You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

   If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

   a. under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

b. under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

   The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

   a. Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
   b. Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

   You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

   a. Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
   b. Accompany the Combined Work with a copy of the GNU GPL and this license document.
   c. For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
   d. Do one of the following:
      0. Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.
      1. Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.
   e. Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

   You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

   a. Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

   b. Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

   The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

   If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

END OF TERMS AND CONDITIONS

GNU Free Documentation License

GNU FREE DOCUMENTATION LICENSE
Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

Terms and Conditions for Copying, Distribution and Modification

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other

material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
   B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
   C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
   D. Preserve all the copyright notices of the Document.
   E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
   F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
   G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
   H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgments" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgments and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you

include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgments", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

END OF TERMS AND CONDITIONS

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.1
or any later version published by the Free Software Foundation;
with the Invariant Sections being list their titles, with the
Front-Cover Texts being list, and with the Back-Cover Texts being list.
A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being *list*"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Indices

Index of Concepts

M

O

P

Q

R

S

T

U

W

Y

Index of Data Types

Index of Functions and Macros

F

G

I

P

S

V

Index of Variables and Constants

Index of Files and Programs

T

Index of Configuration Options

Index of Makefile Targets

Index of Authors

Index of Manual Pages Referenced

A

B

C

D

E

F

G

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z