计 算 机 技 术 开 发 人 员 宝 典 丛 书

8

Sun 核心技术内幕

Sun Solaris 7 流程序设计指南

Sun Solaris 7 Streams Programming Guide

计算机技术开发人员宝典丛书编委会 编

本书配套光盘内容包括与本书配套电子书

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

计 算 机 技 术 开 发 人 员 宝 典 丛 书

8

Sun

核心技术内幕

Sun Solaris 7 流程序设计指南

Sun Solaris 7 Streams Programming Guide

计算机技术开发人员宝典丛书编委会 编

本书配套光盘内容包括与本书配套电子书

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

内 容 简 介

本书是“计算机技术开发人员宝典丛书”Sun 核心技术内幕套书之一，是关于 UNIX 系统通信编程的实用参考书。全书分三个部分共 15 章。第一部分“应用程序编程接口”包括“流”的基础知识、“流”的应用级组件、“流”应用级机制、“流”驱动器和模块接口、“流”管理、管道和队列等 6 章；第二部分“内核接口”包括“流”框架内核级、消息内核级、流驱动程序、配置、多线程“流”和多路复用等 7 章，第三部分“高级主题”包括“流”的终端子系统和调试两章。附录中包括消息类型、“流”实用程序和“流”常见问题解答。

本书按从简单到复杂、从理论到应用的方式编写，既讲述了 Sun Solaris 最新环境中“流”机制的定义和背景，又详细讲解了“流”头、流模块和驱动程序原理及应用机制，并且深入讨论了内核级的流框架、消息、驱动和配置，全书线索清晰，实用性强。本书在讲解流原理机制的同时，还使用了大量的实例进行讲解，便于读者理解和训练。

本书适合于所有对 UNIX 通信编程和通信服务感兴趣的人，尤其对于那些有意深入了解 Sun Solaris 操作系统下“流”编程的用户，这是一本很有价值的参考手册。本书也适用于所有 UNIX 程序员、网络与通信技术人员和高等院校相关专业的师生参考使用。

本书配套光盘内容包括与本书配套的电子书。

系　列　书：计算机技术开发人员宝典丛书（8）
书　　名：Sun 核心技术内幕——Sun Solaris 7 流程序设计指南
文本著作者：计算机技术开发人员宝典丛书编委会　编写
CD 制作者：希望多媒体开发中心
CD 测试者：希望多媒体测试部
责任编辑：陈河南
出版、发行者：北京希望电子出版社
地　　址：北京海淀路 82 号，100080
网址: www.bhp.com.cn
E-mail: lwm@hope.com.cn
电话: 010-62562329,62541992,62637101,62637102，010-62633308,62633309
（发行和技术支持）
010-62613322-215（门市）　010-62531267（编辑部）
经　　销：各地新华书店、软件连锁店

排　　版：希望图书输出中心
CD 生产者：中新联光盘有限公司
文本印刷者：北京广益印刷厂
规格 / 开本：787 毫米×1092 毫米　1/16　17.5 印张　384 千字
版次 / 印次：2000 年 6 月第 1 版　2000 年 6 月第 1 次印刷
印　　数：0001-5000 册
本 版 号：ISBN7-900044 -49-3 /TP • 49
定　　价：38.00 元（1CD，含配套书）

计算机技术开发人员宝典丛书

编 委 会 名 单

序

Sun 公司是 Stanford University Network 三个英文单词首写字母的缩略，中文意为：斯坦福大学网络，创办于 1982 年 2 月。该公司致力于发展网络计算机技术、产品与服务。主要包括大型计算机系统、服务器、工作站操作系统和企业级网络计算机软件。特别是 Unix 工作站/服务器在中国大陆名列第一，市场份额超过 30%。在香港，占有率更高达 52%。Sun 公司的产品和技术主要应用于电信业、制造业、运输业、政府部门、科研教育机构、石油工业和一般商业和分销行业等重要领域。Sun 公司的核心技术有 Ultra SPARC 芯片、Solaris 操作系统以及 Java 与 Jini 技术等。

为满足国内广大从事 Sun 系统应用与开发的技术人员其学习和工作需求，我们与美国斯坦福大学计算机学院和美国软件工程师协会的部分专家共同策划和开发了为培养 21 世纪计算机技术开发人员用的“计算机技术开发人员宝典”丛书，重点介绍 Sun 公司的灵魂技术：Unix 环境下的 Open Windows 和 Solaris 操作系统，该丛书由以下 8 册组成，并将陆续推出以飨读者。

1.《**Open Windows 用户指南**》：主要介绍 OpenWindows 操作系统的操作、系统工具的使用、系统配置、工作区和在线帮助，DeskSet 应用程序，工作区的性质和工作区的纠错等内容。全书由 19 章和 3 个附录构成。其中包括：Solaris 用户环境介绍，文件管理器，文本编辑器，多媒体邮件工具，日历管理器，命令工具、外核工具和控制台窗口，时钟，计算器，性能监视器，打印工具，音频工具，磁带工具，图像工具，瞬态图，图标编辑器，活页夹，定制 Solaris 环境，工具，使用 AnswerBook 软件等；附录介绍了 Solaris 纠错指南、初级用户的访问及词汇表。

2.《**Open Windows 高级用户参考手册**》：该书涵盖了 Solaris 系统软件的基本操作。手册内容均根据具体实例进行讲解，易于理解和掌握。该书由 9 章和 7 个附录构成。主要内容包括：如何注册到 SunOS 和启动 Open Windows，基本的 SunOS 命令，文件和目录的使用，查找文件，管理口令和磁盘存储，使用 vi 编辑程序，收发电子邮件，通过网络进行远程注册、复制文件、执行命令，定制工作环境。在附录中对低版本的 Solaris 系统软件升级、修改键盘、运行联网的应用程序、SPARC-DECnet 网间联网、管理自己的系统和使用 PCMCIA 卡等问题进行了详细的讨论，对用户在使用 Solaris 系统软件过程中遇到的大多数问题提供了答案。非常具有实用参考价值。

3.《**Solaris 通用操作环境用户手册**》：该书全面介绍了 Solaris 通用操作环境的方方面面，以帮助用户了解并有效使用 Solaris 操作环境运行、开发应用程序。全书分 18 章 4 个附录，内容包括：基本技巧、启动桌面会话、帮助、前面板使用说明、文件管理器管理文件、应用程序管理器使用说明、自定义桌面环境、邮件程序使用说明、打印、文本编辑器使用说明、日程使用说明、终端使用说明、图标编辑器使用说明、图像浏览器使用说明、音频使用说明、地址管理器使用说明、进程管理器使用说明、性能监视程序使用说明等。附录给出了桌面的键盘快捷方式、控制本地会话、编辑键序列、地区注释等供参考。

4.《**Solaris 设备与网络接口技术手册**》：该书包括 300 多个项目，详细介绍有关设备、协议、ioctl 请求、STREAM 模块和系统文件等五个方面的问题。本书内容实用、语言简洁，是用户了解 Sun OS 设备与网络技术、打开通往专家之门的金钥匙。通过本书的学习，读者不仅可以具体了解硬件的工作方式、协议的机理、文件系统的组成等信息，还可以提高应用 Sun OS 的综合能力。

5.《**Solaris CDE：高级用户和系统管理员手册**》：该书详细讲述了用于定制 Solaris 公共桌面环境（Common Desktop Environment, CDE）的各项高级任务。全书共分 18 章，其中第 1 章讲述如何配置桌面登录管理员；第 2 章介绍了桌面如何存储和取回会话以及如何定制会话启动；第 3 章描述 Solaris CDE 启动文件、Solaris CDE 启动可能遇到的问题以及启动问题的建议解决方法；第 4 章讲述应用程序管理员如何汇聚应用程序、添加应用程序；第 5 章讲述如何为一个应用程序创建一个注册包；第 6 章讲述高级配置主题；第 7 章讲述如何通过网络分配桌面服务、应用程序和数据；第 8 章讲述如何添加和删除桌面

打印以及如何设定缺省打印机；第 9 章讲述桌面如何通过网络找到应用程序、帮助文件、图标和其他桌面数据；第 10 章介绍动作和数据类型和概念、其如何为应用程序提供用户界面；第 11 章讲述如何使用创建动作应用程序来创建动作和数据类型；第 12 章讲述如何创建动作定义；第 13 章描述如何创建数据类型定义；第 14 章讲述如何使用图标编辑器以及桌面图标的命名约定、大小和搜索路径；第 15 章介绍创建系统级控件和子面板以及其他面板定制；第 16 章讲述如何定制窗口、鼠标按钮捆绑、键盘捆绑以及工作空间管理员菜单；第 17 章描述如何设置应用程序资源以及桌面使用字型和颜色；第 18 章介绍系统进行国际会话时的系统管理任务。该书既适合于 Solaris CDE 高级用户和系统管理员使用，也可供各类软件开发人员参考。

6.《**Solaris 多线程程序设计指南**》：是一本介绍 Solaris 系统中多线程接口编程技术的指导书。全书由 10 章和 3 个附录组成，主要内容包括多线程编程基础介绍，基本线程程序设计，线程属性，在程序中使用同步对象，操作系统编程，接口函数的安全性，编译和调试，增强多线程程序性能的工具，用 Solaris 线程库编程和程序设计原则。附录分别介绍了例子程序——多线程的 grep、Solaris 多线程的例子：barrier C，及多线程安全性级别例子：库接口。本书内容新颖、丰富，实用性强，语言精炼，并附有大量的程序实例。书中既有对多线程编程技术基础知识的全面介绍，又有关于线程库中典型例程的讲解和多线程应用程序的编译、调试、安全性等实际问题的详细介绍。

7.《**Solaris 流程序设计指南**》：一书分三个部分共 15 章。第一部分“应用程序编程接口”包括“流”的概述、“流”的应用级组件、“流”应用级机制、“流”驱动器和模块接口、“流”管理、管道和队列等 6 章；第二部分“内核接口”包括“流”框架内核极、消息内核级、流驱动程序、配置、多线程“流”和多路复用等 7 章，第三部分“高级主题”包括“流”的终端子系统和调试两章。附录中包括消息类型、“流”实用程序和“流”常见问题解答。本书按从简单到复杂、从理论到应用的方式编写，既讲述了 Sun Solaris 最新环境中“流”机制的定义和背景，又详细讲解了“流”头、流模块和驱动程序原理及应用机制，并且深入讨论了内核级的流框架、消息、驱动和配置，全书线索清晰，实用性强。

8.《**i-Planet 网络安全管理指南**》：一书共由 8 章 3 个附录组成。主要内容包括：i-Planet 简介，管理控制台，其他管理工作，i-Planet 用户管理的命令行接口，SSL 服务和证书，管理 i-Planet 的防火墙应用软件，认证和最终用户支持等内容。附录给出了定制 i-Planet HTML 模板文件、可插入的认证模块 API 和第三方软件。本书从 i-Planet 软件的基本内容入手，逐步深入到管理控制台、Web 服务器、授权和加密、防火墙、认证和用户支持等高级内容。全书内容实用、全面，涉及到网络服务器的各个方面，目标明确、条理清晰。

本丛书反映了 90 年代末 21 世纪初国际最新技术的发展，内容定位与国内外技术和产品市场同步，技术内涵高、指导性强，特别针对 SUN 技术用户、应用与开发人员、维护人员、技术支持人员，具有很强的技术参考价值，是以上人员必备的重要技术参考书，也是高等院校相关专业师生教学、自学参考书和国内各图书馆、科研机构重要的馆藏书籍。

藉本丛书出版之际，特别感谢斯坦福大学计算机学院首席教授凯特·哈根特女士，本丛书就是在她的大力帮助和协调下才得以完成。感谢美国软件工程师协会彼德·坎贝尔先生，莎伦·科恩女士，Sun 公司米尔斯·威廉姆斯博士，技术支持部戴维·P·格兰茨博士，由于他们的全力参与和辛勤劳动，本丛书能够及时完稿。特别要感谢王元元、杰克·尼尔森、龙启铭、帕金斯·当、刘大伟、陆卫民、张中民、莎伦·科恩、米尔斯·威廉姆斯、徐建华、普尔森·雷蒙德，由于他们夜以继日的辛勤劳动，本丛书得以及时面市。还要感谢参与本丛书编写的全体专家和技术人员，以及编辑、录排人员和美工设计人员、光盘制作人员等，是他们的加班、加点、忘我的工作，才使本丛书如期付梓出版。

因出版时间紧迫，书中错误在所难免，敬请读者谅解，并请拨冗指正，以期再版时修订。

北京希望电子出版社

2000 年 3 月

目 录

第一部分 应用程序编程接口

第二部分 内核接口

第三部分 高级主题

第一部分　应用程序编程接口

1 “流”的概述

本章是对“流”机制的一个概述，着重介绍“流”机制的一些简单定义和背景，它是后续章节的基础。由于应用程序开发人员和内核级开发人员所关心的“流”接口的子集不同，所以我们对应用级和内核级分开介绍。

- “流”的定义
- 何时用“流”
- “流”是如何工作的——应用程序接口级
- “流”是如何工作的——内核级
- “流”的操作

1.1 什么是“流”

“流”是用于 UNIX 系统通信服务的一种常用的、灵活的编程模型。它为内核以及内核与该 UNIX 系统其他部分之间的字符输入/输出定义标准接口。这种机制应用于多个方面，包括系统调用、内核资源分配和内核例行程序设计等。

你首先创建一些用于标准数据通信服务的模块，然后再通过流对这些模块进行操作。对于应用程序级，模块是动态选择的，并且是互相关联的。但是这种关联不需要内核编码、编译和链接编辑。

“流”为当前所要求的内核服务和驱动程序模块化提供了有效的环境。“流”采用并行分层模型，在网络协议中也能见到这种模型。例如，“流”适用于：

- 实现网络协议
- 开发字符设备驱动
- 开发网络控制器（例如，以太网卡）
- 输入/输出终端服务

“流”（STREAMS）机制的基本单元是流（Stream）。所谓流（Stream），就是用户空间的进程与内核空间的“流”驱动程序之间的全双工的双向数据传输通路。一个流分成三个部分：一个流头（Stream Head），若干个流模块（STREAMS Module）和一个流驱动程序（STREAMS Driver）。

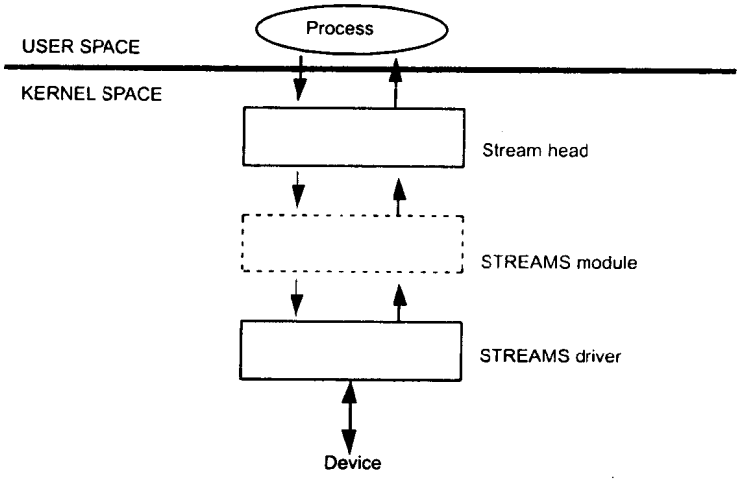

图 1.1　简单的流

1.2　“流”的定义

本书中英文大写的“STREAMS”专指“流”编程模型和设施，小写的“Stream”专指采用“流”模型和设施在用户应用程序与驱动器之间进行全双工通信的事件。在中文版书中，大写字母 STREAMS 用“流”表示，而小写字母“Stream”不加引号，用流表示。

流（Stream）

所谓流，就是用户空间的进程与内核空间的“流”驱动器之间的双向数据传输通路。应用程序通过打开一个流设备创建一个流（参看图 1.1）。

流头（Stream Head）

流头是流最接近用户进程的端点。它是流与用户进程间的接口。当第一次打开流设备时，流仅仅包括流头和“流”驱动器。

模块（Module）

流模块是内核级的例行程序和数据结构的集合。流模块在处理流过它的数据时，像个“黑匣子”。例如，将小写字母转化为大写，或者加入网络路由信息。流模块通常由用户应用程序动态地加载到流当中。关于流模块及其操作细节将在第 10 章详细讨论。

驱动器（Driver）

字符设备驱动器是实现“流（STREAMS）”的接口。流设备驱动器位于流头和所有模块的下方。它可能是外部 I/O 设备，也可能是内部软件驱动，也叫作伪设备驱动器。驱动器负责内核与设备之间的数据传输。这种设备与内核间的接口在 Solaris 7 中叫作设备驱动器接口/驱动器内核接口（Solaris 7 DDI/DKI）。关于这种驱动器与其他 UNIX 内核之间

的关系请参阅《*Writing Device Drivers*》一书，关于设备驱动器的详细解释在第 9 章介绍。

消息（Messages）

所有的 I/O 操作都是由“流”来处理的。在“流”当中，通过流的所有数据都以消息的形式传送。每个流头、流模块和驱动器都有“读方（read-side）”和“写方（write-side）”。当消息从一个模块的“读方”传到下一个模块的“读方”，称为上行流。当消息从一个模块的“写方”传到下一个模块的“写方”，称为下行流。关于内核级的消息操作在本书后面的“消息”一节中讨论。

队列（Queues）

所谓队列，实际上就是消息的容器。每个流头、驱动器和模块都有自己的一对队列，一个是“读”，另一个是“写”。消息通常依照优先级采用先进先出（FIFO）方式顺序进入队列。内核级队列方面的详细内容在本章后面的“队列”一节中讨论。

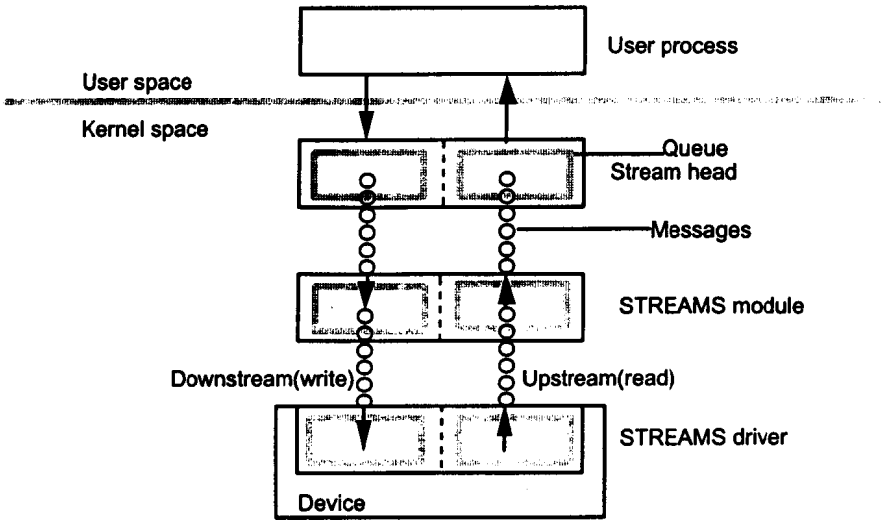

图 1.2 使用队列的消息传递

streamio

为了实现与流设备之间的通信，应用程序进程通过 read（2），write（2），getmsg（2），getpmsg（2），putmsg（2），putpmsg（2）以及 ioctl（2）等命令发送和接收数据。

在命令行状态下，使用 autopush（1M）配置流。在应用程序内部，使用 ioctl（2）配置流，上面 streamio（7I）已提到。

ioctl（2）接口命令主要执行 read（2）和 write（2）对设备驱动不能执行的一些操作，其中包括推入和弹出模块以启动和关闭流、刷新流以及交换一些信号或选项。流中的某些 ioctl（2）命令对整个流操作，而不是仅对模块或驱动器进行操作。streamio（7I）的使用手册中介绍了流 ioctl（2）命令。将在第 4 章详细介绍流的内部通信。

多路复用（Multiplexing）

流的模块化允许一个或多个上面的流将数据复制到一个或多个下面的流。这样的过程就叫作复用（mux）。在“复用”一节中给出了复用器的例子。

轮询（polling）

流中的轮询允许用户进程检测发生在流头部的事件，指定要寻找的事件以及等待该事件的时间。一个应用程序可能需要同多个流打交道。poll（2）系统调用允许应用程序检测发生在一个或多个流头部的事件。详细描述见第3章。

流量控制（Flow control）

流量控制用来规定用户进程，流头、模块以及驱动器之间的消息传输速率。通过使用流量控制，将模块不能处理的数据进行排队，以避免上层模块中的数据溢出。每个模块或驱动器的流控制是局部的，并且是自发的。第8章将详细描述流量控制。

1.3　何时使用“流”

对系统的模块化和可重配置性而言，流机制的框架结构非常有用。例如，通过使用“流”，网络驱动、终端驱动以及图形输入/输出设备驱动都变得很方便。通过模块的推进（加入）和弹出（删除）来创建需要的程序行为。

对于协议级的模块化，“流”机制已经足够了。它是 UNIX 系统 V 的网络支持工具的主要组成部分，因为它实现了网络协议间的通信。

1.4　“流”如何工作——应用程序接口

应用程序首先打开流设备，由它创建流头去访问设备驱动器。流头将来自用户进程的数据封装成“流”消息，然后传递到下层进入内核级。在流头与驱动器间执行各种任务和传递数据时可能需要用到一个或多个流模块。但另一方面，一个流也可能只有流头和驱动器，根本没有流模块。

打开流

对用户应用程序而言，“流”设备与普通的字符型输入/输出设备类似，在文件系统中，它有一个或多个节点，并且也通过系统调用 open（2）命令打开。

文件系统把每个设备看作一个特殊文件。该文件中有一个特殊项——主要设备号，用来标识激活该设备的活动设备驱动器。对特定设备中的每一个事件，将分别有相互分离的次要设备号与之对应，例如串行卡上的特定端口，在视窗应用中使用的虚终端等。

驱动器的不同的次要设备分别对应用户进程和驱动器间的不同连接流。第一个打开的调用创建流，其后打开的调用与该流的文件描述器交换信息。相同的次要设备可能被打开多次，但只创建一个流。

然而，驱动器也可能支持应用程序不区分次要设备的用户进程。这种情况下，驱动器将任意选择一个没有被使用的次要设备供该应用程序使用。这种次要设备的特定应用被称作“克隆”，第9章将介绍“克隆”设备的特性和应用。

一旦设备被打开，用户进程就可通过系统调用 write（2）发送数据给该设备，通过系统调用 read（2）接收来自该设备的数据。通过读和写访问“流”驱动器与传统的字符型输入/输出机制是兼容的。某些特定的流应用程序通过调用 getmsg（2），getpmsg（2），putmsg（2）和 putpmsg（2）在流间传递数据。

关闭流

close（2）用来关闭设备、拆除与被关闭的流最后一次打开时与其他流相关的链接。exit（2）终止用户进程并关闭所有打开的文件。

数据流控制

如果该流使用了流量控制机制，write（2）调用将被阻塞直到流量控制被释放。否则，除非该文件被特殊指明不阻塞。open（2）和 fcntl（2）能够用来控制这种不阻塞行为。

简单流的样例

例 1.1 给出了使用简单流的应用程序代码的例子。这里，用户程序可以通过交互地与通信设备打交道实现两台计算机间的点对点数据传输。写入设备的数据被发送到通信线路上，到达目的地的数据从设备中读出后被复原成原始信息。

例 1.1 简单流

```
#include <sys/fcntl.h>
#include <stdio.h>

main ()
{
  char buf[1024];
  int fd, count;

  if ( (fd = open ("/dev/ttya", O_RDWR) ) < 0) {
   perror ("open failed");
   exit (1);
  }
  while ( (count = read (fd, buf, sizeof (buf) ) ) > 0) {
   if (write (fd, buf, count) != count) {
    perror ("write failed");
    break;
   }
  }
```

```
    exit（0）;
}
```

在该例中，/dev/ttya 表示串行通信设备驱动器。文件被打开后，系统将该设备当作流设备，并将流连接到该设备上。图 1.3 给出了 open（2）调用后的流状态图。

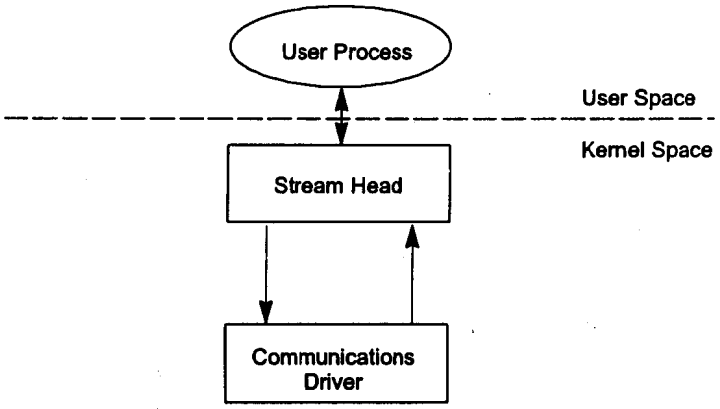

图 1.3　流与通信驱动器的连结

这个例子显示了一个简单的循环。应用程序从通信设备读取数据，然后将输入写回相同的设备，将所有输入返馈回通信线路。该程序每次读 1024 字节，然后写刚刚读出的所有字节。

read（2）返回可用的数据，可能少于 1024 字节。如果读流时，没有可用数据，read（2）将会被阻塞直到数据到达。

注意　*应用程序必须循环 read（2），直到期望的数据全部被读出。负责要求读出所有需要数据的是应用程序开发人员，而不是流设备。*

简单地说，write（2）调用就是试图发送一定数目的数据给/dev/ttya。该设备可以进行流量控制，以防止因设备驱动器数据溢出而耗光系统资源。

1.5　流如何工作——内核级

开发人员往往使用特定的流函数和数据结构来实现流设备驱动和流模块。本节介绍一些基本的内核级流的概念。

流头

流头在用户进程打开一个流设备时创建，它将用户进程的接口调用转化成流消息送给该流。流头也负责将来自流的消息转化成应用程序能处理的格式。一个流头包括一对队列，一个队列将驱动器的消息传给上行流；另一个队列将流消息传给驱动器。这些队列相当于该流的管道，在流头、流模块和驱动器之间传递数据。

流模块

流模块负责处理从流头传到驱动器以及从驱动器传回流头的消息。例如，TCP 模块可能在通过它流向下行流的数据前端加上信息头。不是每个流都需要模块，一个流可能没有也可能有多个模块。流使用模块时执行进栈（推入）和出栈（弹出）操作。每一模块必须提供 open（），close（）和 put（）操作。如果支持流量控制，还必须提供 service（）操作。

像流头一样，每个模块也包含一对队列结构。虽然在实现流量控制时，该模块只是对数据进行排队。图 1.4 给出了模块 A（Au/Ad）和模块 B（Bu/Bd）的队列结构（“u”指上行，“d”指下行）。

两个队列的操作完全相互独立。只有在模块函数指定该程序可共享数据时，上下行的消息和数据才能共享。

在一个模块内，一个队列能够提交相反队列的消息和数据。一个队列可以直接提交给后继模块的队列（临近消息流的方向）。例如，在图 1.4 中，来自模块 A 的上行流队列 Au 可以提交给来自模块 B 的上行流队列 Bu。与此类似，队列 Bd 可以提交给队列 Ad。

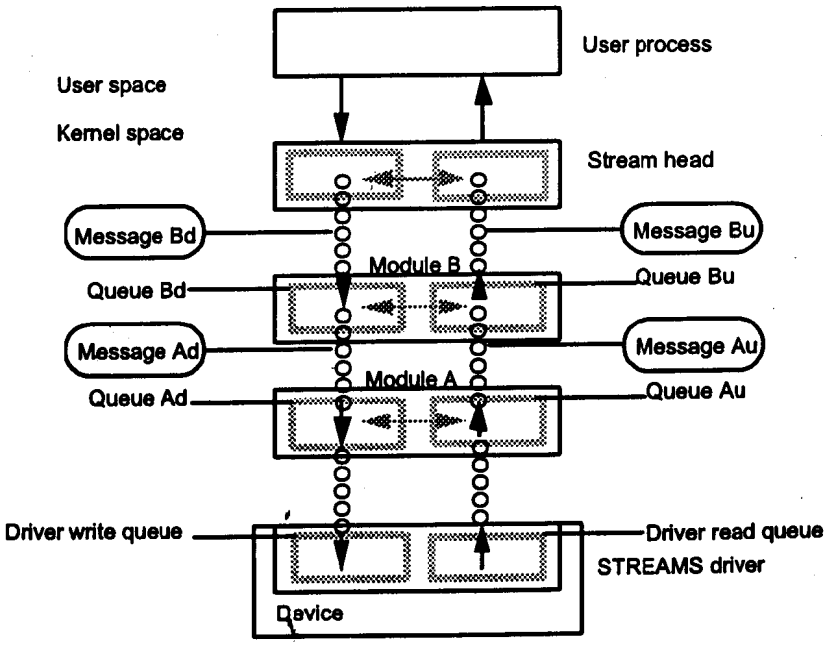

图 1.4 流的细节

流模块中的队列中都要包括消息，处理程序以及私有数据：

消息　　流过模块并能够被模块操作的数据块。

处理程序　　Put 和 Service 是读写消息队列处理数据的两个独立的例行程序。在队列中，put 程序是必需的，它负责将该流中的一个队列的消息传递给另一个队列。service 程序是可选的，它延缓消息的处理。这些程序在上下行流中都能发送消息。而且它们也可以修改所在模块中

的私有数据。

私有数据　　隶属与某一模块的专门数据（例如，状态信息和转移表）。

打开和关闭　　必须提供接入点。当模块被推入流中或者流被重新打开时，open 例程将被激活。当模块被弹出或者流被关闭时，close 例程将被激活。

流模块通常由 I_Push ioctl（2）初始化。否则，如果该流已被 autopush（1M）机制配置或者被重新打开时，在流打开期间，模块会自动推入。流模块由 close 或 I_POP ioctl（2）关闭。

驱动器

“流”设备驱动程序在结构上同流模块和字符设备驱动器相似。驱动器例程的流接口与流模块使用的接口是相同的。例如它们都必须指明 open， close， put 和 service 的入口。

在模块和驱动器之间也有一些重大的差异。

对于驱动器：

- 必须能够处理来自设备的中断。
- 在文件系统中以特殊的字符文件形式出现。
- 使用 open（2）和 close（2）进行初始化和释放。当该设备首次被打开或者重新打开时，open（2）都将被调用，但 close（2）仅仅在流的最后的访问被关上的时候才被调用。

驱动器和模块都能传递信号、错误代码，并根据要求以特定的消息格式将变量值返回给用户进程。

消息

“流”的所有内核级的输入和输出都是基于消息的。流消息由三部分组成：消息标题（msgb（9S））包括与消息实例有关的信息；数据块（datab（9S））包括描述数据的信息；以及数据本身。每对数据块和数据对能被一个或一个以上的消息标题引用。在流模块之间传递的是指向消息的指针。

“流”消息使用两种数据结构（msgb，消息标题，datab，数据块）来描述消息数据。这些数据结构表征消息类型，并指出数据所在位置，另外加上别的信息。消息被连续地调用发送到流中，加入到该流的每个模块或驱动器的程序中。消息能够独立存在，或者被链接在一起形成消息队列的形式。消息和消息队列在流的实用例程中都允许开发者使用和操作。

消息类型

所有的“流”消息都要分配消息类型。这样才能确定它将在模块和驱动器中如何使用，以及被流头怎样处理。消息类型是在消息创建时由流头、驱动器或者模块分配。流头将系统调用 read，write，putmsg 和 putpmsg 等转化成特定的消息类型发送给下行流。它通过复制来自上行流的某些消息类型的内容来应答别的系统调用。

消息排队优先级

有时，一些带有紧急信息的消息，例如中断或警告，必须尽快通过流。为了满足这样

的要求，流在消息排队时使用了优先级以及高优先级消息类型。所有消息都有相关的优先级域。普通（平常）消息的优先级为零，而优先消息有一个大于零的优先范围。高优先级的消息依据它们的消息类型享有更高的优先级，不被流的流量控制阻塞，在所有普通消息之前被处理。

无优先级的普通消息排在队列中正在等待的所有其他消息的后面。优先消息要么是高优先级的，要么是有一定优先范围的。高优先消息将排在队列的头部但必须在已经存在的高优先消息的后面。携带紧急意外数据的有优先范围的消息将排在高优先消息的后面，普通消息的前面。有优先范围的消息将排在所有比自己高和相同优先级的消息的后边，而排在比自己优先级低的消息的前边。图 1.5 给出了消息队列优先级示意图。

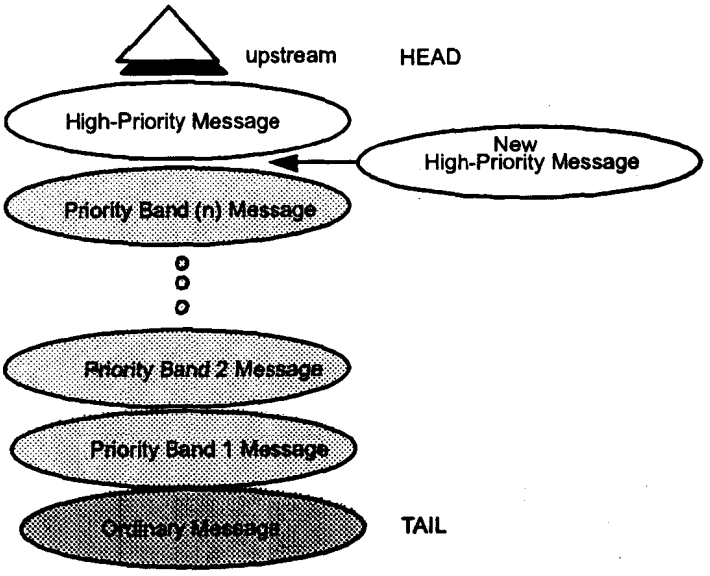

图 1.5 消息的优先级

高优先级消息不能被转化成通常的或有优先范围的消息类型。

某些消息到来时可以是高优先也可以是普通的（例如 M_PCPROTO 和 M_PROTO），这样，当发送消息时，模块或设备驱动器能够在两种优先级中进行选择。

队列

队列是流驱动器或模块与该流的其他部分之间的接口（见 queue（9S））。队列结构应该包含消息以及消息通过模块时指向要处理的流例行程序的指针。数据流模块和驱动器必须明确地将消息放到队列上，例如，就像使用流量控制的时候一样。

每当打开一个驱动器或者推入一个流模块，就要产生相应的一对队列，一个为读，另一个为写。队列总是这样成对分配的。内核日常例程可以访问队列的任一部分。队列的 put 或 service 程序能够把消息加到当前的队列中。如果模块不需要将消息排队，那么它的 put 程序可以调用相邻队列的 put 程序。队列的 service 程序也要对队列上的消息进行处理，通常从队列中移去连续的消息来处理它们，以及调用该流中的下一个模块的 put 程序将消息传递给下一个队列。第 7 章将详细讨论 service 和 put 程序。