



30307

UNIX 系统V 第4版
程序员指南: STREAMS

UNIX® SYSTEM V
RELEASE 4

Programmer's Guide: STREAMS



UNIX Software Operation

電子工業出版社

目 录

第一章 引论	(1)
1.1 本指南介绍	(1)
1.1.1 读者	(1)
1.1.2 组织	(1)
1.1.3 所用的约定	(2)
1.1.4 其它文档	(3)
第二章 STREAMS 概述	(5)
2.1 什么是 STREAMS?	(5)
2.2 基本的流操作	(7)
2.3 STREAMS 的组成部分	(9)
2.3.1 队列	(9)
2.3.2 消息	(10)
2.3.2.1 消息类型	(10)
2.3.2.2 消息入队优先级	(11)
2.3.3 模块	(12)
2.3.4 驱动程序	(13)
2.4 多路复用	(13)
2.5 STREAMS 的长处	(17)
2.5.1 标准化的服务界面	(17)
2.5.2 操作模块	(17)
2.5.2.1 协议可移植性	(18)
2.5.2.2 协议替换	(18)
2.5.2.3 协议迁移	(18)
2.5.2.4 模块可复用性	(19)
第三章 STREAMS 机制	(21)
3.1 STREAMS 机制概述	(21)
3.1.1 STREAMS 系统调用	(21)
3.2 流的构造	(21)
3.2.1 打开一个 STREAMS 设备文件	(23)

3.2.2 创建基于STREAMS的管道	(25)
3.2.3 添加与删除模块	(26)
3.2.4 关闭流	(27)
3.2.5 流构造示例	(27)
3.2.5.1 插入模块	(28)
3.2.5.2 模块和驱动程序控制	(29)
第四章 STREAMS 处理例程	(33)
4.1 put 和 service 过程	(33)
4.1.1 put过程	(33)
4.1.2 service过程	(34)
4.2 异步协议流示例	(34)
4.2.1 读侧处理	(37)
4.2.1.1 驱动程序处理	(37)
4.2.1.2 CHARPROC	(37)
4.2.1.3 CANONPROC	(38)
4.2.2 写侧处理	(38)
4.2.3 分析	(39)
第五章 消息	(41)
5.1 消息	(41)
5.1.1 消息类型	(41)
5.1.2 需加速处理的数据	(42)
5.2 消息结构	(42)
5.2.1 消息链接	(45)
5.2.2 发送/接收消息	(46)
5.2.3 流首处理的控制	(49)
5.2.3.1 读任选项	(50)
5.2.3.2 写偏移量	(50)
5.3 消息队列和消息优先级	(51)
5.3.1 queue结构	(54)
5.3.1.1 使用queue信息	(55)
5.3.1.2 队列标志	(55)
5.3.1.3 equeue结构	(56)
5.3.1.4 qband结构	(56)

5.3.1.5 使用queue和qband信息	(57)
5.3.2 消息处理	(58)
5.3.2.1 流量控制	(60)
5.4 服务界面	(63)
5.4.1 服务界面的长处	(64)
5.4.2 服务界面库示例	(66)
5.4.2.1 访问服务供者	(68)
5.4.2.2 关闭服务供者	(70)
5.4.2.3 把数据发送给服务供者	(70)
5.4.2.4 接收数据	(71)
5.4.2.5 模块服务界面示例	(73)
5.5 消息的分配和释放	(77)
5.5.1 从无缓冲区中恢复	(80)
5.6 扩充的 STREAMS 缓冲区	(82)
第六章 轮询和信号	(85)
6.1 输入/输出轮询	(85)
6.1.1 同步输入/输出	(85)
6.1.2 异步输入/输出	(88)
6.1.3 信号	(89)
6.1.3.1 扩展的信号	(89)
6.2 流作为一个控制终端	(90)
6.2.1 作业控制	(90)
6.2.2 分配和释放	(92)
6.2.3 挂断的流	(92)
6.2.4 挂起信号	(92)
6.2.5 访问控制终端	(92)
第七章 模块和驱动程序概述	(95)
7.1 模块和驱动程序环境	(95)
7.1.1 模块和驱动程序声明	(95)
7.1.1.1 空模块示例	(98)
7.2 模块和驱动程序的 ioctl	(99)
7.2.1 一般ioctl的处理	(100)
7.2.2 L-STRioctl的处理	(102)

7.2.3 透明的ioctl处理	(103)
7.2.4 透明的ioctl消息	(105)
7.2.5 透明的ioctl示例	(106)
7.2.5.1 M_COPYIN示例	(107)
7.2.5.2 M_COPYOUT示例	(109)
7.2.5.3 双向传送示例	(111)
7.2.6 L_LIST ioctl	(115)
7.3 刷清处理	(116)
7.4 驱动程序—核心界面	(120)
7.4.1 设备驱动程序界面和驱动程序—核心界面	(122)
7.4.2 STREAMS界面	(122)
7.5 设计准则	(123)
7.5.1 模块和驱动程序	(123)
7.5.1.1 open/close例程的规则	(124)
7.5.1.2 ioctl的规则	(125)
7.5.1.3 put和service过程的规则	(125)
7.5.2 数据结构	(127)
7.5.2.1 STREAMS数据结构的动态分配	(127)
7.5.3 前导文件	(127)
7.5.4 可访问的符号和函数	(128)
第八章 模块	(131)
8.1 模块	(131)
8.1.1 模块例程	(131)
8.1.2 过滤器模块示例	(134)
8.2 流量控制	(137)
8.3 设计准则	(139)
第九章 驱动程序	(141)
9.1 驱动程序	(141)
9.1.1 驱动程序概述	(141)
9.1.1.1 驱动程序类别	(141)
9.1.1.2 驱动程序配置	(142)
9.1.1.3 编写驱动程序	(142)
9.1.1.4 主设备号和次设备号	(143)

9.1.2	STREAMS驱动程序	(144)
9.1.2.1	打印机驱动程序示例	(146)
9.1.2.2	驱动程序流量控制	(152)
9.2	增殖	(153)
9.3	循环驱动程序	(154)
9.4	设计准则	(161)
第十章	多路复用	(163)
10.1	多路复用	(163)
10.1.1	建造多路转接器	(164)
10.1.2	拆除多路转接器	(169)
10.1.3	通过多路转接器分路数据	(170)
10.2	连接/拆接下层流	(170)
10.2.1	连接下层流	(171)
10.2.2	拆接下层流	(172)
10.3	多路转接器构造实例	(172)
10.4	多路复用驱动程序	(174)
10.4.1	上层写put过程	(177)
10.4.2	上层写service过程	(180)
10.4.3	下层写service过程	(180)
10.4.4	下层读put过程	(181)
10.5	持续的链接	(183)
10.6	设计准则	(186)
第十一章	基于 STREAMS 的管道和 FIFO	(187)
11.1	创建并打开管道和 FIFO	(187)
11.2	访问管道和 FIFO	(188)
11.2.1	从管道或FIFO读	(188)
11.2.2	向管道或FIFO写	(189)
11.2.2.1	长度为0的写	(189)
11.2.2.2	不可分写(atomic write)	(189)
11.2.3	关闭管道或FIFO	(190)
11.3	刷清管道和 FIFO	(190)
11.4	命名的流	(191)
11.4.1	fattach	(191)

11.4.2	<code>fdetach</code>	(192)
11.4.3	<code>isastream</code>	(192)
11.4.4	传递文件描述字	(193)
11.4.5	在远程环境中命名的流	(193)
11.5	唯一的连接	(193)

第十二章 基于 STREAMS 的终端于系统 (197)

12.1	基于 STREAMS 的终端子系统	(197)
12.1.1	线路规程模块	(198)
12.1.1.1	默认设置	(198)
12.1.1.2	数据结构	(199)
12.1.1.3	<code>open/close</code> 例程	(199)
12.1.1.4	读侧处理	(200)
12.1.1.5	写侧处理	(201)
12.1.1.6	在 <code>ldterm</code> 中的 EUC 处理	(201)
12.1.2	<code>termiox(7)</code> 中的支持	(204)
12.1.3	硬件仿真模块	(204)
12.2	基于 STREAMS 的伪终端子系统	(205)
12.2.1	线路规程模块	(206)
12.2.2	伪 <code>tty</code> 仿真模块— <code>PTEM</code>	(206)
12.2.2.1	数据结构	(208)
12.2.2.2	<code>open/close</code> 例程	(209)
12.2.3	远程方式	(209)
12.2.4	分组方式	(210)
12.2.5	伪 <code>tty</code> 驱动程序— <code>ptm</code> 和 <code>pts</code>	(210)
12.2.5.1	<code>grantpt</code>	(213)
12.2.5.2	<code>unlockpt</code>	(213)
12.2.5.3	<code>ptsname</code>	(213)

附录 A STREAMS 数据结构 (215)

A.1	<code>streamtab</code>	(215)
A.2	QUEUE 结构	(215)
A.2.1	<code>queue</code>	(215)
A.2.2	<code>qinit</code>	(216)
A.2.3	<code>module_info</code>	(216)

A.2.4	module_stat	(217)
A.2.5	equeue	(217)
A.2.6	qband	(217)
A.3	消息结构	(218)
A.4	iocblk	(220)
A.5	copyreq	(220)
A.6	copyresp	(221)
A.7	striocctl	(222)
A.8	linkblk	(222)
A.9	stroptions	(223)
附录 B 消息类型		(225)
B.1	消息类型	(225)
B.2	普通消息	(225)
B.3	高优先级消息	(233)
附录 C STREAMS 公用程序		(239)
C.1	STREAMS 公用程序	(239)
C.2	公用程序说明	(239)
adjmsg	调整消息中的字节	(239)
allocb	分配消息和数据块	(240)
backq	得到在给定队列之后的队列的指针	(240)
bcanput	在给定的优先波段内测试流量控制	(240)
buffcall	从allocb失败中恢复	(241)
canput	测试队列中的空间	(241)
copyb	拷贝消息块	(242)
copymsg	拷贝消息	(242)
datamsg	测试消息是否为数据消息	(242)
dupb	复制消息块描述字	(242)
dupmsg	复制消息	(243)
enableok	重新允许队列被调度进行服务	(243)
esballoc	分配消息和数据块	(243)
flushband	刷新在给定的优先波段内的消息	(243)
flushq	刷新队列	(244)
freeb	释放单个消息块	(244)

freemsg	释放消息中所有消息块	(244)
getadmin()	返回指向模块的指针	(245)
getmid	返回模块id	(245)
getq	从队列上得到一个消息	(245)
insq	把一个消息放到队列上指定的地方	(245)
linkb	把两个消息串接成一个	(246)
msgdsize	得到消息中数据字节的数目	(246)
noenable	禁止队列被调度	(246)
OTHERQ	得到配偶队列的指针	(247)
pullupmsg	串接和调整消息中的字节	(247)
putbq	把一个消息放到一个队列的开头	(247)
putctl	放置一控制消息	(247)
putctl1	放置一带有单字节参数的控制消息	(248)
putnext	把一个消息放到下一个队列上	(248)
putq	把一个消息放到队列上	(248)
qenable	启用一队列	(249)
qreply	以相反方向在流上发送消息	(249)
qsize	查出队列上消息的数目	(249)
RD	得到读队列的指针	(250)
rmvb	从消息中删除一消息块	(250)
rmvq	从队列上删除一消息	(250)
splstr	设置处理器级别	(250)
strlog	提交日志消息	(250)
strqget	获得有关队列或该队列波段的信息	(251)
strqset	改变有关一个队列或该队列波段的信息	(252)
testb	检查一个可用缓冲区	(252)
unbufcall	取消一个bufcall请求	(252)
unlinkb	从消息头删除一消息块	(252)
WR	得到写队列的指针	(253)
C.2.1 DK1 界面		(253)
C.3 公用程序例程汇总表		(253)

附录 D 调试

D.1 crash(1M)命令		(256)
D.2 dump 模块示例		(259)

D.3 出错和轨迹登记	(267)
附录 E 配置	(269)
E.1 配置STREAMS模块和驱动程序	(269)
E.1.1 配置示例	(270)
E.1.2 可调参数	(272)
E.2 自动压入设施	(273)
E.2.1 用户界面	(273)
附录 F 手册页	(277)
附录 G 硬件示例	(329)
G.1 硬件示例	(329)
G.1.1 3B2计算机配置机制	(329)
G.2 3B2 基于 STREAMS 的端口驱动程序	(329)
G.2.1 数据结构	(330)
G.2.2 open和close例程	(331)
G.2.3 写put过程	(332)
G.2.4 写service过程	(333)
G.2.5 中断过程	(333)
G.2.6 读service过程	(334)
G.3 3B2 基于 STREAMS 的控制台驱动程序	(335)
G.3.1 数据结构	(336)
G.3.2 open和close例程	(336)
G.3.3 读侧处理	(336)
G.3.3.1 中断级处理	(336)
G.3.3.2 服务过程处理	(337)
G.3.4 写侧处理	(337)
G.3.5 匿名方式	(338)
G.4 3B2 基于 STREAMS 的 XT 驱动程序	(338)
G.4.1 数据结构	(343)
G.4.2 open处理	(345)
G.4.3 close处理	(346)
G.4.4 数据流程	(346)
G.4.5 读侧处理	(347)

G.4.6 写侧处理	(348)
G.4.7 多路实用	(350)
G.4.8 流量控制	(350)
G.4.8.1 STREAMS流量控制	(350)
G.4.8.2 XT驱动程序协议流量控制	(351)
G.4.9 扫描	(351)
G.4.10 循环冗余检验	(351)
G.4.11 编码	(351)
G.5 扩展的 STREAMS 缓冲区	(352)
词汇表	(355)

图和表

图 2-1: 简单流	(6)
图 2-2: 基于 STREAMS 的管道	(6)
图 2-3: 到通信驱动程序的流	(8)
图 2-4: 消息	(10)
图 2-5: 消息队列中的消息	(11)
图 2-6: 有关流的细节	(12)
图 2-7: 多对一多路转换器	(14)
图 2-8: 一对多多路转换器	(14)
图 2-9: 多对多多路转换器	(15)
图 2-10: Internet 多路转换流	(15)
图 2-11: X.25 多路转换流	(16)
图 2-12: 协议模块可移植性	(18)
图 2-13: 协议迁移	(19)
图 2-14: 模块可复用性	(20)
图 3-1: 逆流和顺流流的构造	(22)
图 3-2: 流中队列关系	(23)
图 3-3: 打开的基于 STREAMS 的驱动程序	(24)
图 3-4: 创建基于 STREAMS 的管道	(26)
图 3-5: 大小写转换程序模块	(29)
图 4-1: 空转流配置示例	(35)
图 4-2: 可操作流示例	(36)
图 4-3: 模块 put 和 service 过程	(37)

图 5-1: 消息的构成和链接	(45)
图 5-2: 队列中的消息排序	(51)
图 5-3: 具有一个优先波段的消息顺序	(51)
图 5-4: 非 EFT 系统中的数据结构链接	(58)
图 5-5: 流量控制	(60)
图 5-6: 协议替换	(64)
图 5-7: 服务界面	(65)
图 7-1: 刷清流的写侧	(118)
图 7-2: 刷清流的读侧	(119)
图 7-3: 影响驱动程序的界面	(121)
图 9-1: 设备驱动程序流	(145)
图 9-2: 循环流	(154)
图 10-1: 协议多路转接器	(164)
图 10-2: 连接前	(165)
图 10-3: 第一次连接后的 IP 多路转接器	(166)
图 10-4: IP 多路转接器	(167)
图 10-5: TP 多路转接器	(168)
图 10-6: 连接前的 Internet 多路转接器	(173)
图 10-7: 连接后的 Internet 多路转接器	(174)
图 10-8: MUXdriver 和 Driver1 和 open()	(183)
图 10-9: 在 I-PLINK 以后的多路转接器	(184)
图 10-10: 其他用户打开一个 MUXdriver	(185)
图 11-1: 把模块压入基于 STREAMS 的管道	(188)
图 11-2: 服务方建立一个管道	(194)
图 11-3: 进程 X 和 Y 打开 /usr/ioserv	(194)
图 12-1: 基于 STREAMS 的终端子系统	(197)
图 12-2: 伪 tty 子系统体系结构	(207)
图 B-1: M-PROTO 和 M-PCPRCTO 消息结构	(229)
图 D-1: 出错和轨迹登记	(268)
图 G-1: 基于 STREAMS 的 XT 驱动程序(连接前)	(339)
图 G-2: 基于 STREAMS 的 XT 驱动程序(连接后)	(340)
图 G-3: 基于 STREAMS 的 XT 驱动程序	(341)
图 G-4: starlan 上基于 STREAMS 的 XT 驱动程序	(342)
图 G-5: 基于 STREAMS 的 XT 驱动程序数据流	(347)
图 G-6: 3B2 上的 UNIX I/O	(353)
图 G-7: 386 上的 UNIX I/O	(354)

第一章 引论

1.1 本指南介绍

本指南向开发者提供在用户层和内核层使用 STREAMS 机制的信息。

在 UNIX System V R3 中加入了 STREAMS，这是为了扩展字符输入/输出(I/O)机制以及支持通信服务的开发。

STREAMS 向开发者提供一整套函数，一组公用程序例程以及加速软件设计和实现的设施。

1.1.1 读者

本指南主要面向网络和系统程序员，他们在 UNIX 系统通信服务的用户层和内核层使用 STREAMS 机制。

希望本指南的读者已具备有关 UNIX 系统、编程、网络和数据通信方面的知识。

1.1.2 组织

本指南分成若干章，每一章讨论一个独立的专题。第二、三和四章是介绍性的，已经熟悉 STREAMS 概念和功能的读者可以跳过这几章。

- 第一章，“引论”，描述本指南的组织 and 宗旨。它也规定了本指南所面向的读者以及希望他们应该具备的背景知识。
- 第二章，“STREAMS概述”，描述了STREAMS的概貌和长处。
- 第三章，“STREAMS机制”，描述了构造、使用和拆除流的基本操作。这些操作用 `open(2)`、`close(2)`、`read(2)`、`write(2)` 和 `ioctl(2)` 完成。
- 第四章，“STREAMS处理例程”，提供了STREAMS put和service例程的概貌。
- 第五章，“消息”，讨论了STREAMS消息，它们的结构、链接、排队以及与其它STREAMS组成部分的界面。
- 第六章，“轮询和信号”，描述了STREAMS怎样允许用户进程控制、管理和轮询流，以便有效地利用系统资源。
- 第七章，“模块和驱动程序概述”，描述了STREAMS模块和驱动程序环境、`ioctl`、例程、声明、刷清处理、驱动程序-内核界面，也提供了模块和驱动程序的一般性的设计准则。
- 第八章，“模块”，提供了有关模块构造和功能方面的信息。
- 第九章，“驱动程序”，讨论了STREAMS驱动程序、驱动程序流量控制、刷清处理以及增殖处理。

- 第十章,“多路复用”,描述STREAMS多路复用设施。
- 第十一章,“基于STREAMS的管道和FIFO”,提供有关基于STREAMS的管道和FIFO的创建、写入、读取、关闭以及唯一的连接方面的信息。
- 第十二章,“基于STREAMS的终端子系统”,讨论了基于STREAMS的终端和伪终端子系统。
- 附录A,“STREAMS数据结构”,汇集了STREAMS模块和驱动程序共同使用的数据结构。
- 附录B,“消息类型”,描述了STREAMS消息及其使用。
- 附录C,“STREAMS公用程序”,描述STREAMS公用程序例程和它们的用法。
- 附录D,“调试”,向开发者提供调试手段。
- 附录E,“配置”,描述如何把模块和驱动程序配置到UNIX系统中,如何调整参数以及自动压入设施。
- 附录F,“手册页”,提供了与STREAMS有关的手册页。
- 附录G,“硬件示例”,提供了用于STREAMS环境下,适合于某些硬件类型,比如AT&T 3B2的信息。
- “词汇表”定义了STREAMS独有的术语。

1.1.3 所用的约定

在本指南中,词“STREAMS”表示机制,而词“流”(stream)表示用户应用程序和驱动程序之间的通路。在基于STREAMS的管道部分,“流”表示在内核中,内核与一个或多个用户进程间的数据传输通路。

给出了一些示例,其目的是为了强调STREAMS中最重要和公共的能力。这些示例并不很完备,并且为了简单起见,引用了虚构的驱动程序和模块。

在本指南中,系统调用、STREAMS公用程序例程、前导文件以及数据结构都以等线体字给出。

变量名、指针和参数以斜体字给出。当提及示例中所独有的例程、字段和结构名时也以斜体字给出。

说明和短小的例子以等线体给出。

```

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

Data structure formats are also shown if screens.

```

小心: 本符号用于说明系统、应用程序、进程以及硬件的某个部分等,可能存在的损害和危险。

注意：本符号用于强调要点、提供附加的信息以及列举参考文档和命令。

1.1.4 其它文档

虽然本指南是一种有助于开发 STREAMS 应用的基本工具，但是鼓励读者从其它文档中获得更多的信息，这些信息包括 STREAMS 使用的系统调用(手册页第 2 节)和 STREAMS 公用程序(手册页 1M 节)。STREAMS 特定的输入-输出控制(`ioctl`)调用在 `streamio(7)` 中提供。STREAMS 模块和驱动程序在手册页第 7 节中提供。《System V 界面定义》第三版的某些扩充条目中也对 STREAMS 作了描述。

有关 AT&T UNIX System V R4.0 详细书目的清单，请参见有关 R4.0 版的《产品概览和总索引》。

第二章 STREAMS 概述

2.1 什么是 STREAMS?

“STREAMS”是一种通用、灵活的设施和用于开发 UNIX 系统通信服务的一组工具。它在很大范围内支持多种服务的实现，从完整的网络协议包到单独一个设备驱动程序。STREAMS 定义用于内核中的字符输入/输出，以及内核和 UNIX 系统其余部分之间的标准界面。相关的机制是简单而且开放的。它由一组系统调用、内核资源和内核例程组成。

这种标准界面和机制使得高性能的网络服务和它们的组成成分能够实现模块化，可以移植开发和容易集成。STREAMS 并不局限于任何特殊的网络体系结构。STREAMS 用户界面与 `open`、`close`、`read`、`write` 和 `ioctl` 等用户层字符 I/O 函数向上兼容。本章后面将详细地讨论 STREAMS 的长处。

“流”是在内核空间中的 STREAMS 驱动程序与用户空间中的进程之间的一种全双工处理和数据传输通路(参见图 2-1)。在内核中，流通过连接流首、驱动程序以及它们之间的零个或多个模块而构成。“流首”是流最靠近用户进程的那一端。由流上用户层进程发出的所有系统调用都由流首处理。

管道也是基于 STREAMS 的。基于 STREAMS 的管道(参见图 2-2)是内核中一种全双工(双向)的数据传输通路。它实现了内核与一个或几个用户进程间的连接，也具有基于 STREAMS 设备的优点。

“STREAMS 驱动程序”可以是提供外部 I/O 设备服务的一种设备驱动程序；也可以是一种软件驱动程序，通常这种驱动程序称为伪设备驱动程序。这种驱动程序主要处理内核与设备间的数据传输。除了 STREAMS 机制使用的数据结构与该设备理解的数据结构间的转换之外，它很少或根本不处理别的数据。

“STREAMS 模块”提供在流的数据流程上实施的处理功能。模块定义成一组用于处理数据、状态以及控制信息的内核层例程和数据结构。数据处理可能涉及修改数据表示的方式，在数据上添加或删除前导或后续信息以及/或者打包/拆包数据。状态和控制信息包括信号和输入/输出控制信息。每一个模块是自我封闭的，而且除了它的两个相邻的成分外，在功能上它与该流中所有其它成分相隔绝。模块通过传递消息与它的相邻成分通信。模块不是 STREAMS 中的必要成分，但除了基于 STREAMS 的管道(这里只需要流首)外，驱动程序是一定需要的。

在流首和驱动程序之间可以插入一个或多个模块，以便在流首和驱动程序间传递消息时对其进行中间处理。STREAMS 模块由用户进程在流中动态地互连。创建这种连接不需要内核编程、汇编或连接编辑。