Glossary

autopush

A STREAMS mechanism that enables a pre-specified list of modules to be pushed automatically onto the Stream when a STREAMS device is opened. This mechanism is used only for administrative purposes.

back-enable

To enable (by STREAMS) a preceding blocked queue's `service` procedure when STREAMS determines that a succeeding queue has reached its low watermark.

blocked

A queue's `service` procedure that cannot be enabled due to flow control.

clone device

A STREAMS device that returns an unused major/minor device when initially opened, rather than requiring the minor device to be specified by name in the `open`(2) call.

close routine

A procedure that is called when a module is popped from a Stream or when a driver is closed.

controlling Stream

A Stream above the multiplexing driver used to establish the lower connections. Multiplexed Stream configurations are maintained through the controlling Stream to a multiplexing driver.

DDI

Device Driver Interface. An interface that facilitates driver portability across different UNIX system versions on SPARC® hardware.

DKI

Driver–Kernel Interface. An interface between the UNIX system kernel and different types of drivers. It consists of a set of driver-defined functions that are called by the kernel. These functions are entry points into a driver.

downstream

A direction of data flow going from the Stream head towards a driver. Also called write-side and output side.

device driver

A Stream component whose principle functions are handling an associated physical device and transforming data and information between the external interface and Stream.

driver

A module that forms the Stream end. It can be a device driver or a pseudo-device driver. It is a required component in STREAMS (except in STREAMS-based pipe mechanism), and physically identical to a module. It typically handles data transfer between the kernel and a device and does little or no processing of data.

enable

A term used to describe scheduling of a queue's `service` procedure.

FIFO

First-In-First-Out. A term for named pipes. This term is also used in queue scheduling.

flow control

A STREAMS mechanism that regulates the rate of message transfer within a Stream and from user space into a Stream.

hardware emulation module

A module required when the terminal line discipline is on a Stream but there is no terminal driver at the end of a Stream. This module understands all `ioctls` necessary to support terminal semantics specified by `termio`(7) and `termios`(7).

input side

A direction of data flow going from a driver towards the Stream head. Also called read-side and upstream.

line discipline

A STREAMS module that performs `termio`(7) canonical and non-canonical processing. It shares some `termio`(7) processing with a driver in a STREAMS terminal subsystem.

lower Stream

A Stream connected below a multiplexer pseudo-device driver, by means of an `I_LINK` or `I_PLINK` `ioctl`. The far end of a lower Stream terminates at a device driver or another multiplexer driver.

master driver

A STREAMS-based device supported by the pseudo-terminal subsystem. It is the controlling part of the pseudo-terminal subsystem (also called `ptm`).

message

One or more linked message blocks. A message is referenced by its first message block and its type is defined by the message type of that block.

message block

A triplet consisting of a data buffer and associated control structures, an `msgb` structure and a `datab` structure. It carries data or information, as identified by its message type, in a Stream.

message queue

A linked list of zero or more messages connected together.

message type

A defined set of values identifying the contents of a message.

module

A defined set of kernel-level routines and data structures used to process data, status and control information on a Stream. It is an optional element, but there can be many modules in one Stream. It consists of a pair of queues (read queue and write queue), and it communicates to other components in a Stream by passing messages.

multiplexer

A STREAMS mechanism that allows messages to be routed among multiple Streams in the kernel. A multiplexing configuration includes at least one multiplexing pseudo-device driver connected to one or more upper Streams and one or more lower Streams.

named Stream

A Stream, typically a pipe, with a name associated with it via a call to `fattach`(3C) (that is, a mount operation). This is different from a named pipe (FIFO) in two ways: a named pipe (FIFO) is unidirectional while a named Stream is bidirectional; a named Stream need not refer to a pipe but can be another type of a Stream.

open routine

A procedure in each STREAMS driver and module called by STREAMS on each `open`(2) system call made on the Stream. A module's open procedure is also called when the module is pushed.

packet mode

A feature supported by the STREAMS-based pseudo-terminal subsystem. It is used to inform a process on the master side when state changes occur on the slave side of a pseudo-TTY. It is enabled by pushing a module called `pckt` on the master side.

persistent link

A connection below a multiplexer that can exist without having an open controlling Stream associated with it.

pipe

Same as a STREAMS-based pipe.

pop

A term used when a module that is immediately below the Stream head is removed.

pseudo-device driver

A software driver, not directly associated with a physical device, that performs functions internal to a Stream such as a multiplexer or log driver.

pseudo-terminal subsystem

A user interface identical to a terminal subsystem except that there is a process in a place of a hardware device. It consists of at least a master device, slave device, line discipline module, and hardware emulation module.

push

A term used when a module is inserted in a Stream immediately below the Stream head.

pushable module

A module put between the Stream head and driver. It performs intermediate transformations on messages flowing between the Stream head and driver. A driver is a non-pushable module.

put procedure

A routine in a module or driver associated with a queue which receives messages from the preceding queue. It is the single entry point into a queue from a preceding queue. It may perform processing on the message and will then generally either queue the message for subsequent processing by this queue's `service` procedure, or will pass the message to the `put` procedure of the following queue.

queue

A data structure that contains status information, a pointer to routines processing messages, and pointers for administering a Stream. It typically contains pointers to a `put` and `service` procedure, a message queue, and private data.

read-side

A direction of data flow going from a driver towards the Stream head. Also called upstream and input side.

read queue

A message queue in a module or driver containing messages moving upstream. Associated with the `read`(2) system call and input from a driver.

remote mode

A feature available with the pseudo-terminal subsystem. It is used for applications that perform the canonical and echoing functions normally done by the line discipline module and tty driver. It enables applications on the master side to turn off the canonical processing.

SAD

A STREAMS Administrative Driver that provides an interface to the autopush mechanism.

schedule

To place a queue on the internal list of queues which will subsequently have their `service` procedure called by the STREAMS scheduler. STREAMS scheduling is independent of the Solaris process scheduling.

service interface

A set of primitives that define a service at the boundary between a service user and a service provider and the rules (typically represented by a state machine) for allowable sequences of the primitives across the boundary. At a Stream/user boundary, the primitives are typically contained in the control part of a message; within a Stream, in `M_PROTO` or `M_PCPROTO` message blocks.

service procedure

A routine in module or driver associated with a queue that receives messages queued for it by the `put` procedure of that queue. The procedure is called by the STREAMS scheduler. It may perform processing on the message and generally passes the message to the `put` procedure of the following queue.

service provider

An entity in a service interface that responds to request primitives from the service user with response and event primitives.

service user

An entity in a service interface that generates request primitives for the service provider and consumes response and event primitives.

slave driver

A STREAMS-based device supported by the pseudo-terminal subsystem. It is also called `pts` and works with a line discipline module and hardware emulation module to provide an interface to a user process.

standard pipe

A mechanism for a unidirectional flow of data between two processes where data written by one process become data read by the other process.

Stream

A kernel aggregate created by connecting STREAMS components, resulting from an application of the STREAMS mechanism. The primary components are the Stream head, the driver, and zero or more pushable modules between the Stream head and driver.

STREAMS-based pipe

A mechanism used for bidirectional data transfer implemented using STREAMS, and sharing the properties of STREAMS-based devices.

Stream end

A Stream component furthest from the user process, containing a driver.

Stream head

A Stream component closest to the user process. It provides the interface between the Stream and the user process.

STREAMS

A kernel mechanism that provides the framework for network services and data communication. It defines interface standards for character input/output within the kernel, and between the kernel and user level. The STREAMS mechanism comprises integral functions, utility routines, kernel facilities, and a set of structures.

TTY driver

A STREAMS-based device used in a terminal subsystem.

upper Stream

A Stream that terminates above a multiplexer. The beginning of an upper Stream originates at the Stream head or another multiplexer driver.

upstream

A direction of data flow going from a driver towards the Stream head. Also called read-side and input side.

watermark

A limit value used in flow control. Each queue has a high watermark and a low watermark. The high watermark value indicates the upper limit related to the number of bytes contained on the queue. When the queued character reaches its high watermark, STREAMS causes another queue that attempts to send a message to this queue to become blocked. When the characters in this queue are reduced to the low watermark value, the other queue will be unblocked by STREAMS.

write queue

A message queue in a module or driver containing messages moving downstream. Associated with the `write`(2) system call and output from a user process.

write-side

A direction of data flow going from the Stream head towards a driver. Also called downstream and output side.