NAME

srv – service queued messages

SYNOPSIS

#include <sys/types.h>
#include <sys/stream.h>
#include <sys/stropts.h>
#include <sys/ddi.h>
#include <sys/sunddi.h>

int *prefix***rsrv(queue_t** **q***);** /* **read side** */

int *prefix***wsrv(queue_t** **q***);** /* **write side** */

INTERFACE LEVEL

Architecture independent level 1 (DDI/DKI). This entry point is **required** for STREAMS.

ARGUMENTS

q Pointer to the **queue**(9S) structure.

DESCRIPTION

The optional service (**srv( )**) routine may be included in a STREAMS module or driver for many possible reasons, including:

- to provide greater control over the flow of messages in a stream
- to make it possible to defer the processing of some messages to avoid depleting system resources
- to combine small messages into larger ones, or break large messages into smaller ones
- to recover from resource allocation failure. A module's or driver's **put**(9E) routine can test for the availability of a resource, and if it is not available, enqueue the message for later processing by the **srv** routine.

A message is first passed to a module's or driver's **put**(9E) routine, which may or may not do some processing. It must then either:

- Pass the message to the next stream component with **putnext**(9F).
- If a **srv** routine has been included, it may call **putq**(9F) to place the message on the queue

Once a message has been enqueued, the STREAMS scheduler controls the service routine's invocation. The scheduler calls the service routines in FIFO order. The scheduler cannot guarantee a maximum delay **srv** routine to be called except that it will happen before any user level process are run.

Every stream component (stream head, module or driver) has limit values it uses to implement flow control. Each component should check the tunable high and low water marks to stop and restart the flow of message processing. Flow control limits apply only between two adjacent components with **srv** routines.

STREAMS messages can be defined to have up to 256 different priorities to support requirements for multiple bands of data flow. At a minimum, a stream must distinguish between normal (priority zero) messages and high priority messages (such as **M_IOCACK**). High priority messages are always placed at the head of the **srv** routine's queue, after any other enqueued high priority messages. Next are messages from all included priority bands, which are enqueued in decreasing order of priority. Each priority band has its own flow control limits. If a flow controlled band is stopped, all lower priority bands are also stopped.

Once the STREAMS scheduler calls a **srv** routine, it must process all messages on its queue. The following steps are general guidelines for processing messages. Keep in mind that many of the details of how a **srv** routine should be written depend of the implementation, the direction of flow (upstream or downstream), and whether it is for a module or a driver.

1. Use **getq**(9F) to get the next enqueued message.
2. If the message is high priority, process (if appropriate) and pass to the next stream component with **putnext**(9F).
3. If it is not a high priority message (and therefore subject to flow control), attempt to send it to the next stream component with a **srv** routine. Use **bcanputnext**(9F) to determine if this can be done.
4. If the message cannot be passed, put it back on the queue with **putbq**(9F). If it can be passed, process (if appropriate) and pass with **putnext( )**.

RETURN VALUES

Ignored.

SEE ALSO

put(9E), **bcanput**(9F), **bcanputnext**(9F), **canput**(9F), **canputnext**(9F), **getq**(9F), **nulldev**(9F) **putbq**(9F), **putnext**(9F), **putq**(9F), **queue**(9S)

Writing Device Drivers
STREAMS Programming Guide

WARNINGS

Each stream module must specify a read and a write service (**srv( )**) routine. If a service routine is not needed (because the **put( )** routine processes all messages), a **NULL** pointer should be placed in module's **qinit**(9S) structure. Do not use **nulldev**(9F) instead of the **NULL** pointer. Use of **nulldev**(9F) for a **srv( )** routine may result in flow control errors.