NAME

ng_socket - netgraph socket node type

SYNOPSIS

#include <sys/types.h>
#include <netgraph/ng_socket.h>

DESCRIPTION

A **socket** node is both a BSD socket and a netgraph node. The **ng_socket** node type allows user-mode processes to participate in the kernel netgraph(4) networking subsystem using the BSD socket interface. The process must have root privileges to be able to create netgraph sockets however once created, any process that has one may use it.

A new **ng_socket** node is created by creating a new socket of type NG_CONTROL in the protocol family PF_NETGRAPH, using the socket(2) system call. Any control messages received by the node and not having a cookie value of NGM_SOCKET_COOKIE are received by the process, using recvfrom(2); the socket address argument is a struct sockaddr_ng containing the sender's netgraph address. Conversely, control messages can be sent to any node by calling sendto(2), supplying the recipient's address in a struct sockaddr_ng. The bind(2) system call may be used to assign a global netgraph name to the node.

To transmit and receive netgraph data packets, a NG_DATA socket must also be created using socket(2) and associated with a **ng_socket** node. NG_DATA sockets do not automatically have nodes associated with them; they are bound to a specific node via the connect(2) system call. The address argument is the netgraph address of the **ng_socket** node already created. Once a data socket is associated with a node, any data packets received by the node are read using recvfrom(2) and any packets to be sent out from the node are written using sendto(2). In the case of data sockets, the struct sockaddr_ng contains the name of the *hook* on which the data was received or should be sent.

As a special case, to allow netgraph data sockets to be used as stdin or stdout on naive programs, a sendto(2) with a NULL sockaddr pointer, a send(2) or a write(2) will succeed in the case where there is exactly ONE hook attached to the socket node, (and thus the path is unambiguous).

There is a user library that simplifies using netgraph sockets; see netgraph(3).

HOOKS

This node type supports hooks with arbitrary names (as long as they are unique) and always accepts hook connection requests.

CONTROL MESSAGES

This node type supports the generic control messages, plus the following:

NGM_SOCK_CMD_NOLINGER
When the last hook is removed from this node, it will shut down as if it had received a NGM_SHUTDOWN message. Attempts to access the sockets associated will return ENOTCONN.

NGM_SOCK_CMD_LINGER
This is the default mode. When the last hook is removed, the node will continue to exist, ready to accept new hooks until it is explicitly shut down.

All other messages with neither the NGM_SOCKET_COOKIE or NGM_GENERIC_COOKIE will be passed unaltered up the NG_CONTROL socket.

SHUTDOWN

This node type shuts down and disappears when both the associated NG_CONTROL and NG_DATA sockets have been closed, or a NGM_SHUTDOWN control message is received. In the latter case, attempts to write to the still-open sockets will return ENOTCONN. If the NGM_SOCK_CMD_NOLINGER message has been received, closure of the last hook will also initiate a shutdown of the node.

SEE ALSO

socket(2), netgraph(3), netgraph(4), ng_ksocket(4), ngctl(8)

HISTORY

The **ng_socket** node type was implemented in FreeBSD 4.0.

AUTHORS

Julian Elischer <*julian@FreeBSD.org*>

BUGS

It is not possible to reject the connection of a hook, though any data received on that hook can certainly be ignored.

The controlling process is not notified of all events that an in-kernel node would be notified of, e.g. a new hook, or hook removal. Some node-initiated messages should be defined for this purpose (to be sent up the control socket).