NAME

ng_ksocket - kernel socket netgraph node type

SYNOPSIS

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

DESCRIPTION

A **ksocket** node is both a netgraph node and a BSD socket. The **ng_ksocket** node type allows one to open a socket inside the kernel and have it appear as a Netgraph node. The **ng_ksocket** node type is the reverse of the socket node type (see ng_socket(4)): whereas the socket node type enables the user-level manipulation (via a socket) of what is normally a kernel-level entity (the associated Netgraph node), the **ng_ksocket** node type enables the kernel-level manipulation (via a Netgraph node) of what is normally a user-level entity (the associated socket).

A **ng_ksocket** node allows at most one hook connection. Connecting to the node is equivalent to opening the associated socket. The name given to the hook determines what kind of socket the node will open (see below). When the hook is disconnected and/or the node is shutdown, the associated socket is closed.

HOOKS

This node type supports a single hook connection at a time. The name of the hook must be of the form *<family>/<type>/<proto>*, where the *family*, *type*, and *proto* are the decimal equivalent of the same arguments to socket(2). Alternately, aliases for the commonly used values are accepted as well. For example inet/dgram/udp is a more readable but equivalent version of 2/2/17.

Data received into socket is sent out via hook. Data received on hook is sent out from socket, if the latter is connected (an NGM_KSOCKET_CONNECT was sent to node before). If socket is not connected, destination *struct sockaddr* must be supplied in an mbuf tag with cookie NGM_KSOCKET_COOKIE and type NG_KSOCKET_TAG_SOCKADDR attached to data. Otherwise **ng_ksocket** will return ENOTCONN to sender.

CONTROL MESSAGES

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

NGM_KSOCKET_BIND (**bind**)
: This functions exactly like the bind(2) system call. The *struct sockaddr* socket address parameter should be supplied as an argument.

NGM_KSOCKET_LISTEN (**listen**)

This functions exactly like the listen(2) system call. The backlog parameter (a single 32 bit int) should be supplied as an argument.

NGM_KSOCKET_CONNECT (**connect**)
This functions exactly like the connect(2) system call. The *struct sockaddr* destination address parameter should be supplied as an argument.

NGM_KSOCKET_ACCEPT (**accept**)
Equivalent to the accept(2) system call on a non-blocking socket. If there is a pending connection on the queue, a new socket and a corresponding cloned node are created. Returned are the cloned node's ID and a peer name (as *struct sockaddr*). If there are no pending connections, this control message returns nothing, and a connected node will receive the above message asynchronously, when a connection is established.

A cloned node supports a single hook with an arbitrary name. If not connected, a node disappears when its parent node is destroyed. Once connected, it becomes an independent node.

NGM_KSOCKET_GETNAME (**getname**)
Equivalent to the getsockname(2) system call. The name is returned as a *struct sockaddr* in the arguments field of the reply.

NGM_KSOCKET_GETPEERNAME (**getpeername**)
Equivalent to the getpeername(2) system call. The name is returned as a *struct sockaddr* in the arguments field of the reply.

NGM_KSOCKET_SETOPT (**setopt**)
Equivalent to the setsockopt(2) system call, except that the option name, level, and value are passed in a *struct ng_ksocket_sockopt*.

NGM_KSOCKET_GETOPT (**getopt**)
Equivalent to the getsockopt(2) system call, except that the option is passed in a *struct ng_ksocket_sockopt*. When sending this command, the value field should be empty; upon return, it will contain the retrieved value.

ASCII FORM CONTROL MESSAGES

For control messages that pass a *struct sockaddr* in the argument field, the normal ASCII equivalent of the C structure is an acceptable form. For the PF_INET and PF_LOCAL address families, a more convenient form is also used, which is the protocol family name, followed by a slash, followed by the actual address. For PF_INET, the address is an IP address followed by an optional colon and port number. For PF_LOCAL, the address is the pathname as a doubly quoted string.

Examples:

PF_LOCAL local/"/tmp/foo.socket"

PF_INET inet/192.168.1.1:1234

Other { family=16 len=16 data=[0x70 0x00 0x01 0x23] }

For control messages that pass a *struct ng_ksocket_sockopt*, the normal ASCII form for that structure is used. In the future, more convenient encoding of the more common socket options may be supported.

Setting socket options example:

Set FIB 2 for a socket (SOL_SOCKET, SO_SETFIB):

```
setopt { level=0xffff name=0x1014 data=[ 2 ] }
```

SHUTDOWN

This node shuts down upon receipt of a NGM_SHUTDOWN control message, or when the hook is disconnected. Shutdown of the node closes the associated socket.

SEE ALSO

socket(2), netgraph(4), ng_socket(4), ngctl(8), mbuf_tags(9), socket(9)

HISTORY

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

AUTHORS

Archie Cobbs <*archie@FreeBSD.org*>