NAME

socket – create an endpoint for communication

SYNOPSIS

cc [*flag* ...] *file* ... **–lsocket –lnsl** [*library* ...]

#include <sys/types.h>
#include <sys/socket.h>

int socket(int *domain***, int** *type***, int** *protocol***);**

MT-LEVEL

Safe

DESCRIPTION

socket( ) creates an endpoint for communication and returns a descriptor.

The *domain* parameter specifies a communications domain within which communication will take place; this selects the protocol family which should be used. The protocol family generally is the same as the address family for the addresses supplied in later operations on the socket. These families are defined in the include file **<sys/socket.h>**. There must be an entry in the **netconfig**(4) file for at least each protocol family and type required. If *protocol* has been specified, but no exact match for the tuplet family, type, protocol is found, then the first entry containing the specified family and type with zero for protocol will be used. The currently understood formats are:

PF_UNIX UNIX system internal protocols

PF_INET ARPA Internet protocols

The socket has the indicated *type*, which specifies the communication semantics. Currently defined types are:

SOCK_STREAM
SOCK_DGRAM
SOCK_RAW
SOCK_SEQPACKET
SOCK_RDM

A **SOCK_STREAM** type provides sequenced, reliable, two-way connection-based byte streams. An out-of-band data transmission mechanism may be supported. A **SOCK_DGRAM** socket supports datagrams (connectionless, unreliable messages of a fixed (typically small) maximum length). A **SOCK_SEQPACKET** socket may provide a sequenced, reliable, two-way connection-based data transmission path for datagrams of fixed maximum length; a consumer may be required to read an entire packet with each read system call. This facility is protocol specific, and presently not implemented for any protocol family. **SOCK_RAW** sockets provide access to internal network interfaces. The types **SOCK_RAW**, which is available only to the super-user, and **SOCK_RDM**, for which no implementation currently exists, are not described here.

protocol specifies a particular protocol to be used with the socket. Normally only a single protocol exists to support a particular socket type within a given protocol family. However, multiple protocols may exist, in which case a particular protocol must be specified in this manner. The protocol number to use is particular to the “communication domain” in which communication is to take place. If a protocol is specified by the caller, then it will be packaged into a socket level option request and sent to the underlying protocol layers.

Sockets of type **SOCK_STREAM** are full-duplex byte streams, similar to pipes. A stream socket must be in a *connected* state before any data may be sent or received on it. A connection to another socket is created with a **connect**(3N) call. Once connected, data may be transferred using **read**(2) and **write**(2) calls or some variant of the **send**(3N) and **recv**(3N) calls. When a session has been completed, a **close**(2) may be performed. Out-of-band data may also be transmitted as described on the **send**(3N) manual page and received as described on the **recv**(3N) manual page.

The communications protocols used to implement a **SOCK_STREAM** insure that data is not lost or duplicated. If a piece of data for which the peer protocol has buffer space cannot be successfully transmitted within a reasonable length of time, then the connection is considered broken and calls will indicate an error with −1 returns and with **ETIMEDOUT** as the specific code in the global variable **errno**. The protocols optionally keep sockets "warm" by forcing transmissions roughly every minute in the absence of other activity. An error is then indicated if no response can be elicited on an otherwise idle connection for a extended period (for instance 5 minutes). A **SIGPIPE** signal is raised if a process sends on a broken stream; this causes naive processes, which do not handle the signal, to exit.

SOCK_SEQPACKET sockets employ the same system calls as **SOCK_STREAM** sockets. The only difference is that **read**(2) calls will return only the amount of data requested, and any remaining in the arriving packet will be discarded.

SOCK_DGRAM and **SOCK_RAW** sockets allow datagrams to be sent to correspondents named in **sendto**(3N) calls. Datagrams are generally received with **recvfrom**(3N), which returns the next datagram with its return address.

An **fcntl**(2) call can be used to specify a process group to receive a **SIGURG** signal when the out-of-band data arrives. It may also enable non-blocking I/O and asynchronous notification of I/O events with **SIGIO** signals.

The operation of sockets is controlled by socket level *options*. These options are defined in the file **<sys/socket.h>**. **setsockopt**(3N) and **getsockopt**(3N) are used to set and get options, respectively.

RETURN VALUES

A −1 is returned if an error occurs. Otherwise the return value is a descriptor referencing the socket.

ERRORS

The **socket( )** call fails if:

EACCES Permission to create a socket of the specified type and/or protocol is denied.

EMFILE The per-process descriptor table is full.

ENOMEM Insufficient user memory is available.

ENOSR There were insufficient STREAMS resources available to complete the operation.

EPROTONOSUPPORT The protocol type or the specified protocol is not supported within this domain.

SEE ALSO

close(2), **fcntl**(2), **ioctl**(2), **read**(2), **write**(2), **accept**(3N), **bind**(3N), **connect**(3N), **getsockname**(3N), **getsockopt**(3N), **listen**(3N), **recv**(3N), **recvfrom**(3N), **setsockopt**(3N), **send**(3N), **shutdown**(3N), **socketpair**(3N)