NAME

send, **sendto**, **sendmsg**, **sendmmsg** - send message(s) from a socket

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/socket.h>

ssize_t
send(*int s*, *const void *msg*, *size_t len*, *int flags*);

ssize_t
sendto(*int s*, *const void *msg*, *size_t len*, *int flags*, *const struct sockaddr *to*, *socklen_t tolen*);

ssize_t
sendmsg(*int s*, *const struct msghdr *msg*, *int flags*);

ssize_t
sendmmsg(*int s*, *struct mmsghdr * restrict msgvec*, *size_t vlen*, *int flags*);

DESCRIPTION

The **send**() and **sendmmsg**() functions, and **sendto**() and **sendmsg**() system calls are used to transmit one or more messages (with the **sendmmsg**() call) to another socket. The **send**() function may be used only when the socket is in a *connected* state. The functions **sendto**(), **sendmsg**() and **sendmmsg**() may be used at any time if the socket is connectionless-mode. If the socket is connection-mode, the protocol must support implied connect (currently tcp(4) is the only protocol with support) or the socket must be in a connected state before use.

The address of the target is given by *to* with *tolen* specifying its size, or the equivalent *msg_name* and *msg_namelen* in *struct msghdr*. If the socket is in a connected state, the target address passed to **sendto**(), **sendmsg**() or **sendmmsg**() is ignored. The length of the message is given by *len*. If the message is too long to pass atomically through the underlying protocol, the error EMSGSIZE is returned, and the message is not transmitted.

The **sendmmsg**() function sends multiple messages at a call. They are given by the *msgvec* vector along with *vlen* specifying the vector size. The number of octets sent per each message is placed in the *msg_len* field of each processed element of the vector after transmission.

No indication of failure to deliver is implicit in a **send**(). Locally detected errors are indicated by a return value of -1.

If no messages space is available at the socket to hold the message to be transmitted, then **send**() normally blocks, unless the socket has been placed in non-blocking I/O mode. The select(2) system call may be used to determine when it is possible to send more data.

The *flags* argument may include one or more of the following:

```
#define   MSG_OOB              0x00001 /* process out-of-band data */
#define   MSG_DONTROUTE        0x00004 /* bypass routing, use direct interface */
#define   MSG_EOR              0x00008 /* data completes record */
#define   MSG_DONTWAIT         0x00080 /* do not block */
#define   MSG_EOF              0x00100 /* data completes transaction */
#define   MSG_NOSIGNAL 0x20000 /* do not generate SIGPIPE on EOF */
```

The flag MSG_OOB is used to send "out-of-band" data on sockets that support this notion (e.g. SOCK_STREAM); the underlying protocol must also support "out-of-band" data. MSG_EOR is used to indicate a record mark for protocols which support the concept. The MSG_DONTWAIT flag request the call to return when it would block otherwise. MSG_EOF requests that the sender side of a socket be shut down, and that an appropriate indication be sent at the end of the specified data; this flag is only implemented for SOCK_STREAM sockets in the PF_INET protocol family. MSG_DONTROUTE is usually used only by diagnostic or routing programs. MSG_NOSIGNAL is used to prevent SIGPIPE generation when writing a socket that may be closed.

See recv(2) for a description of the *msghdr* structure and the *mmsghdr* structure.

RETURN VALUES

The **send**(), **sendto**() and **sendmsg**() calls return the number of octets sent. The **sendmmsg**() call returns the number of messages sent. If an error occurred a value of -1 is returned.

ERRORS

The **send**() and **sendmmsg**() functions and **sendto**() and **sendmsg**() system calls fail if:

[EBADF]	An invalid descriptor was specified.
[EACCES]	The destination address is a broadcast address, and SO_BROADCAST has not been set on the socket.
[ENOTCONN]	The socket is connection-mode but is not connected.

[ENOTSOCK]	The argument *s* is not a socket.
[EFAULT]	An invalid user space address was specified for an argument.
[EMSGSIZE]	The socket requires that message be sent atomically, and the size of the message to be sent made this impossible.
[EAGAIN]	The socket is marked non-blocking, or MSG_DONTWAIT is specified, and the requested operation would block.
[ENOBUFS]	The system was unable to allocate an internal buffer. The operation may succeed when buffers become available.
[ENOBUFS]	The output queue for a network interface was full. This generally indicates that the interface has stopped sending, but may be caused by transient congestion.
[EHOSTUNREACH]	The remote host was unreachable.
[EISCONN]	A destination address was specified and the socket is already connected.
[ECONNREFUSED]	The socket received an ICMP destination unreachable message from the last message sent. This typically means that the receiver is not listening on the remote port.
[EHOSTDOWN]	The remote host was down.
[ENETDOWN]	The remote network was down.
[EADDRNOTAVAIL]	The process using a SOCK_RAW socket was jailed and the source address specified in the IP header did not match the IP address bound to the prison.
[EPIPE]	The socket is unable to send anymore data (SBS_CANTSENDMORE has been set on the socket). This typically means that the socket is not connected.

SEE ALSO

connect(2), fcntl(2), getsockopt(2), recv(2), select(2), socket(2), write(2), CMSG_DATA(3)

HISTORY

The **send**() function appeared in 4.2BSD. The **sendmmsg**() function appeared in FreeBSD 11.0.

BUGS

Because **sendmsg**() does not necessarily block until the data has been transferred, it is possible to transfer an open file descriptor across an AF_UNIX domain socket (see recv(2)), then **close**() it before it has actually been sent, the result being that the receiver gets a closed file descriptor. It is left to the application to implement an acknowledgment mechanism to prevent this from happening.