NAME

recv, **recvfrom**, **recvmsg**, **recvmmsg** - receive message(s) from a socket

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/socket.h>

ssize_t
recv(*int s, void *buf, size_t len, int flags*);

ssize_t
recvfrom(*int s, void *buf, size_t len, int flags, struct sockaddr * restrict from,*
*socklen_t * restrict fromlen*);

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

ssize_t
recvmmsg(*int s, struct mmsghdr * restrict msgvec, size_t vlen, int flags,*
*const struct timespec * restrict timeout*);

DESCRIPTION

The **recvfrom**(), **recvmsg**(), and **recvmmsg**() system calls are used to receive messages from a socket, and may be used to receive data on a socket whether or not it is connection-oriented.

If *from* is not a null pointer and the socket is not connection-oriented, the source address of the message is filled in. The *fromlen* argument is a value-result argument, initialized to the size of the buffer associated with *from*, and modified on return to indicate the actual size of the address stored there.

The **recv**() function is normally used only on a *connected* socket (see connect(2)) and is identical to **recvfrom**() with a null pointer passed as its *from* argument.

The **recvmmsg**() function is used to receive multiple messages at a call. Their number is supplied by *vlen*. The messages are placed in the buffers described by *msgvec* vector, after reception. The size of each received message is placed in the *msg_len* field of each element of the vector. If *timeout* is NULL the call blocks until the data is available for each supplied message buffer. Otherwise it waits for data for the specified amount of time. If the timeout expired and there is no data received, a value 0 is returned. The ppoll(2) system call is used to implement the timeout mechanism, before first receive is

performed.

The **recv**(), **recvfrom**() and **recvmsg**() return the length of the message on successful completion, whereas **recvmmsg**() returns the number of received messages. If a message is too long to fit in the supplied buffer, excess bytes may be discarded depending on the type of socket the message is received from (see socket(2)).

If no messages are available at the socket, the receive call waits for a message to arrive, unless the socket is non-blocking (see fcntl(2)) in which case the value -1 is returned and the global variable *errno* is set to EAGAIN. The receive calls except **recvmmsg**() normally return any data available, up to the requested amount, rather than waiting for receipt of the full amount requested; this behavior is affected by the socket-level options SO_RCVLOWAT and SO_RCVTIMEO described in getsockopt(2). The **recvmmsg**() function implements this behaviour for each message in the vector.

The select(2) system call may be used to determine when more data arrives.

The *flags* argument to a **recv**() function is formed by *or*'ing one or more of the values:

MSG_OOB	process out-of-band data
MSG_PEEK	peek at incoming message
MSG_TRUNC	return real packet or datagram length
MSG_WAITALL	wait for full request or error
MSG_DONTWAIT	do not block
MSG_CMSG_CLOEXEC	set received fds close-on-exec
MSG_WAITFORONE	do not block after receiving the first message (only for **recvmmsg**())

The MSG_OOB flag requests receipt of out-of-band data that would not be received in the normal data stream. Some protocols place expedited data at the head of the normal data queue, and thus this flag cannot be used with such protocols. The MSG_PEEK flag causes the receive operation to return data from the beginning of the receive queue without removing that data from the queue. Thus, a subsequent receive call will return the same data. The MSG_TRUNC flag causes the receive operation to return the full length of the packet or datagram even if larger than provided buffer. The flag is supported on SOCK_DGRAM sockets for AF_INET , AF_INET6 and AF_UNIX families. The MSG_WAITALL flag requests that the operation block until the full request is satisfied. However, the call may still return less data than requested if a signal is caught, an error or disconnect occurs, or the next data to be received is of a different type than that returned. The MSG_DONTWAIT flag requests the call to return when it would block otherwise. If no data is available, *errno* is set to EAGAIN. This flag is not available in ANSI X3.159-1989 ("ANSI C89") or ISO/IEC 9899:1999 ("ISO C99") compilation mode. The MSG_WAITFORONE flag sets MSG_DONTWAIT after the first message has been received. This

flag is only relevant for **recvmmsg**().

The **recvmsg**() system call uses a *msghdr* structure to minimize the number of directly supplied arguments. This structure has the following form, as defined in <*sys/socket.h*>:

```
struct msghdr {
        void            *msg_name;      /* optional address */
        socklen_t       msg_namelen;    /* size of address */
        struct iovec    *msg_iov;       /* scatter/gather array */
        int             msg_iovlen;     /* # elements in msg_iov */
        void            *msg_control;   /* ancillary data, see below */
        socklen_t       msg_controllen; /* ancillary data buffer len */
        int             msg_flags;      /* flags on received message */
};
```

Here *msg_name* and *msg_namelen* specify the source address if the socket is unconnected; *msg_name* may be given as a null pointer if no names are desired or required. The *msg_iov* and *msg_iovlen* arguments describe scatter gather locations, as discussed in read(2). The *msg_control* argument, which has length *msg_controllen*, points to a buffer for other protocol control related messages or other miscellaneous ancillary data. The messages are of the form:

```
struct cmsghdr {
        socklen_t       cmsg_len;       /* data byte count, including hdr */
        int             cmsg_level;     /* originating protocol */
        int             cmsg_type;      /* protocol-specific type */
/* followed by
        u_char          cmsg_data[]; */
};
```

As an example, the SO_TIMESTAMP socket option returns a reception timestamp for UDP packets.

With AF_UNIX domain sockets, ancillary data can be used to pass file descriptors and process credentials. See unix(4) for details.

The *msg_flags* field is set on return according to the message received. MSG_EOR indicates end-of-record; the data returned completed a record (generally used with sockets of type SOCK_SEQPACKET). MSG_TRUNC indicates that the trailing portion of a datagram was discarded because the datagram was larger than the buffer supplied. MSG_CTRUNC indicates that some control data were discarded due to lack of space in the buffer for ancillary data. MSG_OOB is returned to indicate that expedited or out-of-band data were received.

The **recvmmsg**() system call uses the *mmsghdr* structure, defined as follows in the <*sys/socket.h*> header:

```
struct mmsghdr {
        struct msghdr   msg_hdr;        /* message header */
        ssize_t         msg_len;/* message length */
};
```

On data reception the *msg_len* field is updated to the length of the received message.

RETURN VALUES

These calls except **recvmmsg**() return the number of bytes received. **recvmmsg**() returns the number of messages received. A value of -1 is returned if an error occurred.

ERRORS

The calls fail if:

[EBADF] The argument *s* is an invalid descriptor.

[ECONNRESET] The remote socket end is forcibly closed.

[ENOTCONN] The socket is associated with a connection-oriented protocol and has not been connected (see connect(2) and accept(2)).

[ENOTSOCK] The argument *s* does not refer to a socket.

[EMFILE] The **recvmsg**() system call was used to receive rights (file descriptors) that were in flight on the connection. However, the receiving program did not have enough free file descriptor slots to accept them. In this case the descriptors are closed, with pending data either discarded in the case of the unreliable datagram protocol or preserved in the case of a reliable protocol. The pending data can be retrieved with another call to **recvmsg**().

[EMSGSIZE] The *msg_iovlen* member of the *msghdr* structure pointed to by *msg* is less than or equal to 0, or is greater than *IOV_MAX*.

[EAGAIN] The socket is marked non-blocking and the receive operation would block, or a receive timeout had been set and the timeout expired before data were received.

[EINTR] The receive was interrupted by delivery of a signal before any data were

available.

[EFAULT] The receive buffer pointer(s) point outside the process’s address space.

SEE ALSO

fcntl(2), getsockopt(2), read(2), select(2), socket(2), CMSG_DATA(3), unix(4)

HISTORY

The **recv**() function appeared in 4.2BSD. The **recvmmsg**() function appeared in FreeBSD 11.0.