NAME

recv, recvfrom, recvmsg – receive a message from a socket

SYNOPSIS

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

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

int recvfrom(int *s***, void *****buf***, int** *len***, unsigned int** *flags* **struct sockaddr *** *from***, int *** *fromlen***);**

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

DESCRIPTION

The **recvfrom** and **recvmsg** 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 non-nil, and the socket is not connection-oriented, the source address of the message is filled in. *Fromlen* is a value-result parameter, 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** call is normally used only on a *connected* socket (see **connect**(2)) and is identical to **recvfrom** with a nil *from* parameter. As it is redundant, it may not be supported in future releases.

All three routines return the length of the message on successful completion. 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 nonblocking (see **fcntl**(2)) in which case the value −1 is returned and the external variable *errno* set to **EWOULDBLOCK**. The receive calls 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 **select**(2) call may be used to determine when more data arrive.

The *flags* argument to a recv call 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_WAITALL
wait for full request or error

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_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 **recvmsg** call uses a *msghdr* structure to minimize the number of directly supplied

parameters. This structure has the following form, as defined in *sys/socket.h*:

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

Here *msg_name* and *msg_namelen* specify the destination address if the socket is unconnected; *msg_name* may be given as a null pointer if no names are desired or required. *Msg_iov* and *msg_iovlen* describe scatter gather locations, as discussed in **read**(2). *Msg_control*, 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 {
        u_int    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, one could use this to learn of changes in the data-stream in XNS/SPP, or in ISO, to obtain user-connection-request data by requesting a recvmsg with no data buffer provided immediately after an **accept** call.

Open file descriptors are now passed as ancillary data for **AF_UNIX** domain sockets, with *cmsg_level* set to **SOL_SOCKET** and *cmsg_type* set to **SCM_RIGHTS**.

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.

RETURN VALUES

These calls return the number of bytes received, or −1 if an error occurred.

ERRORS

EBADF The argument *s* is an invalid descriptor.

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.

EWOULDBLOCK
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.

CONFORMING TO

4.4BSD (these function calls first appeared in 4.2BSD).

SEE ALSO

fcntl(2), **read**(2), **select**(2), **getsockopt**(2), **socket**(2)