NAME

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

SYNOPSIS

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

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

int recv(int *s*, **char** **buf*, **int** *len*, **int** *flags***);**

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

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

MT-LEVEL

Safe

DESCRIPTION

recv( ), **recvfrom( )**, and **recvmsg( )** are used to receive messages from another socket. **recv( )** may be used only on a *connected* socket (see **connect**(3N)), while **recvfrom( )** and **recvmsg( )** may be used to receive data on a socket whether it is in a connected state or not. *s* is a socket created with **socket**(3N).

If *from* is not a NULL pointer, 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 length of the message is returned. 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**(3N)).

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 **-1** is returned with the external variable **errno** set to **EWOULDBLOCK**.

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

The *flags* parameter is formed by ORing one or more of the following:

MSG_OOB Read any "out-of-band" data present on the socket rather than the regular "in-band" data.

MSG_PEEK "Peek" at the data present on the socket; the data is returned, but not consumed, so that a subsequent receive operation will see the same data.

The **recvmsg( )** call uses a **msghdr** structure to minimize the number of directly supplied parameters. This structure is defined in **<sys/socket.h>** and includes the following members:

```
caddr_t       msg_name;         /* optional address */
int           msg_namelen;      /* size of address */
struct iovec  *msg_iov;         /* scatter/gather array */
int           msg_iovlen;       /* # elements in msg_iov */
caddr_t       msg_accrights;    /* access rights sent/received */
int           msg_accrightslen;
```

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. The **msg_iov** and **msg_iovlen** describe the scatter-gather locations, as described in **read**(2). A buffer to receive any access rights sent along with the message is specified in **msg_accrights**, which has length **msg_accrightslen**.

RETURN VALUES

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

ERRORS

The calls fail if:

EBADF	*s* is an invalid file descriptor.
EINTR	The operation was interrupted by delivery of a signal before any data was available to be received.
EIO	An I/O error occurred while reading from or writing to the file system.
ENOMEM	There was insufficient user memory available for the operation to complete.
ENOSR	There were insufficient STREAMS resources available for the operation to complete.
ENOTSOCK	*s* is not a socket.
ESTALE	A stale NFS file handle exists.
EWOULDBLOCK	The socket is marked non-blocking and the requested operation would block.

SEE ALSO

fcntl(2), **ioctl**(2), **read**(2), **connect**(3N), **getsockopt**(3N), **send**(3N), **socket**(3N)