sendmsg

Sends a Message to a Connected or Unconnected Socket

Portability: UNIX compatible

SYNOPSIS

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

int sendmsg(int s struct msghdr *mh, int flags)
```

DESCRIPTION

sendmsg sends a message to an unconnected or connected socket **s** . **mh** points to a structure containing further parameters. The definition of the **msghdr** structure is in the **<sys/socket.h>** header file. The elements of this structure are as follows:

msg_name
points to a structure in which **sendmsg** stores the source address of the message that is being snet. This field can be **NULL** if the socket **s** is connected, or if the application does not require information on the source address.

msg_namelen
is the length of the buffer pointed to by **msg_name** .

msg_iov
points to an array of **struct iovec** similar to that used by **readv** .

msg_iovlen
is the number of elements in the array pointed to by **msg_iov** .

msg_accrights
is ignored for **AF_INET** .

msg_accrightslen
is ignored for **AF_INET** .

flags consists of the following:

MSG_OOB
requests that message buffers be sent as out-of-band data.

MSG_DONTROUTE
bypasses routing; uses the network portion of the destination address to select a network interface.

RETURN VALUE

If **sendmsg** succeeds, it returns the length of the message. Otherwise, it returns a **-1** , and sets **errno** to indicate the type of error.

CAUTION

sendmsg is an atomic operation. With UDP, no more than one datagram can be read per call. If you are using datagram sockets, make sure that there is enough buffer space in the I/O vector to contain an incoming datagram.

PORTABILITY

mh is commonly documented as **struct msghdr mh[]** , implying that the call operates on an array of these structures. In reality, only one structure is modified by the call. For the purposes of clarity, this manual documents **mh** in a different way, but the implementation is the same. **sendmsg** is portable to other environments, including most UNIX systems, that implement BSD sockets.

EXAMPLE

In this example, **sendmsg** is used to transmit banking transactions.

```
#include <sys/types.h>
#include <sys/uio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>

#include "transdef.h"                    /* application header file  */

/* This routine writes banking transactions to one of several         */
/* regional servers (chosen based on the contents of the transaction, */
/* not based on the location of the local host). "s" is a datagram    */
/* socket. "head" and "trail" are application level transaction       */
/* header and trailer components. "trans" is the body of the          */
/* transaction. Reliability must be ensured by the caller (because    */
/* datagrams do not provide it). The server receives all three        */
/* parts as a single datagram.                                        */
puttrans(int s, struct header *head, struct record *trans,
         struct trailer *trail)
{
   int rc;

      /* socket address for server                                    */
   struct sockaddr_in dest;

      /* will contain information about the remote host               */
   struct hostent *host;
   char fullname[64];
```

```
   /* Will point to the segments of the (noncontiguous)            */
   /* outgoing message.                                            */
struct iovec iov[3];

   /* This structure contains parameter information for sendmsg.   */
struct msghdr mh;
   /* Choose destination host from region field of the data        */
   /* header. Then find its IP address.                            */
strcpy(fullname,head->region);
strcat(fullname,".mis.bank1.com");
host = gethostbyname(fullname);
if (host==NULL) {
    printf("Host %s is unknown.\n",fullname);
    return -1;
}

/* Fill in socket address for the server. We assume a        */
/* standard port is used.                                                  */
memset(&dest,'\0',sizeof(dest));
dest.sin_family = AF_INET;
memcpy(&dest.sin_addr,host->h_addr,sizeof(dest.sin_addr));
dest.sin_port = htons(TRANSACTION_SERVER);

   /* Specify the components of the message in an "iovec".   */
iov[0] .iov_base = (caddr_t)head;
iov[0] .iov_len = sizeof(struct header);
iov[1] .iov_base = (caddr_t)trans;
iov[1] .iov_len = sizeof(struct record);
iov[2] .iov_base = (caddr_t)trail;
iov[2] .iov_len = sizeof(struct trailer);

   /* The message header contains parameters for sendmsg.    */
mh.msg_name = (caddr_t) &dest;
mh.msg_namelen = sizeof(dest);
mh.msg_iov = iov;
mh.msg_iovlen = 3;
mh.msg_accrights = NULL;            /* irrelevant to AF_INET */
mh.msg_accrightslen = 0;            /* irrelevant to AF_INET */

rc = sendmsg(s, &mh, 0);            /* no flags used         */
if (rc == -1) {
   perror("sendmsg failed");
   return -1;
}
```

```
   return 0;
}
```

RELATED FUNCTIONS

connect , **getsockopt** , **ioctl** , **recv** , **recvfrom** , **recvmsg** , **send** , **sendto** , **setsockopt**

Chapter Contents Previous Next Top of Page