NAME

msg, msgenab, msgdisab, send, sendw, recv, recvw, msgstat, msgctl − send and receive messages

SYNOPSIS

# include <sys/ipcomm.h>
msgenab ( )
msgdisab ( )

send (buf, size, topid, type)
sendw (buf, size, topid, type)
char *buf;

recv (buf, size, &mstructp, type)
recvw (buf, size, &mstructp, type)
char *buf;
struct mstruct mstructp;

msgstat (&mstat, sizeof (mstat), pid)
struct mstat mstat;

msgctl (pid, command, arg)

DESCRIPTION

A process that has enabled message reception has a message queue on which are placed, in order of arrival, messages sent to it by other processes. The process actually receives a message's contents by requesting a message from the queue. A process may send a message to any other process that has enabled message reception, as long as the receiver does not have an excessive number of messages pending on its queue.

From assembly language, the *function* argument specifies the request type.

0 Message reception is disabled; messages may no longer be sent to the process. Depending on the *type*, any message(s) still on the queue are either discarded or returned to the sender. No other arguments are used for this kind of request.

1 Enable message reception. No messages may be sent to the process until this is done. No other arguments are used in this kind of request. Message reception remains enabled across *exec*, but not across *fork*.

2 Send a message to another process. If the system's message buffers are temporarily full, return is immediate. (Conditional Send)

3 Send a message to another process. This is as above, except that execution may be suspended until there is sufficient buffer space to send the message. (Unconditional Send)

4 Receive the first message on the queue of the requested *type*. Return immediately if no such message exists. (Conditional Receive)

5 Receive a message as above, except that execution may be suspended until a suitable message is placed on the queue, if one is not already available. (Unconditional Receive)

6 Request a count of the number of messages allowed and actual number of messages queued for the process numbered *pid*.

7 Set control variables in message queue header as defined by *command*. At present, only available command is *setmqlen* which sets maximum number of messages allowed by process numbered *pid*.

The *buf* argument is the address of the buffer that, when sending, contains the message to be sent, or, when receiving, is where the message is to be placed. The number of bytes to be sent or received should be in r0. Currently, messages may be from 0 to 212 bytes in length. If, when receiving, the length of the message exceeds the requested number of bytes, the message is truncated. In any event, the number of bytes actually sent or received is returned in r0.

When a message is being sent, *arg3* should contain the processid of the receiving process. When receiving a message, *arg3* should be the address of a structure of type mstruct.

The *type* argument is used by a sender to assign a type number (1 to 128) to a message. By convention, types 1 to 63 imply that an acknowledgement message is desired; types 64 to 128 imply no acknowledgement is necessary; type 128 is an acknowledgement message. If a process disables messages (or exits) with any messages still on its queue, those of type 1 to 63 are changed to type 128 and, if possible, returned to the sender; those of type 64 to 128 are discarded.

When receiving messages, a process may request *type* 0, indicating that the first message on the queue is to be retrieved, or a *type* from 1 to 128, indicating that the first message on the queue of the requested *type* is to be received. In either case, the message's actual type is returned in the second word of the structure provided by the user *arg3*.

From C, *msgenab* and *msgdisab* enable and disable message reception, respectively. *Msgstat* returns message status in terms of actual and maximum allowed message queue lengths. *Msgctl* allows modification of the maximum number of messages parameter. All return zero when successful.

The *send*, *sendw*, *recv*, and *recvw* functions perform conditional send, unconditional send, conditional receive, and unconditional receive operations, respectively. All return the number of bytes actually sent or received, as appropriate. The format of *ipcomm.h* is as follows:

```
/*              %W%             */

/*
 * Interprocess Communication Control Structures
 */

#ifdef KERNEL
/*
 * common flags
 */

#define IP_PERM         03                              /* scope permission mask */
#define IP_ANY 0                                /* system scope */
#define IP_UID  01                              /* userid scope */
#define IP_GID  02                              /* groupid scope */
#define IP_QWANT        0100                    /* entry in msg queue wanted */
#define IP_WANTED       0200                    /* resource is desired */

struct ipaword
{               char            ip_flag;
                char            ip_id; };

/*
 * message control
 */

#define PMSG   5                                /* message sleep priority */
#define MAXMLEN         212                             /* max message length in bytes */
#define MAXMSGDEF       10                              /* default max number unreceived msgs per
#define MAXMSGL         20                              /*max limit to be set by msgctl*/
```

```
#define MSGIO  02                                   /* tell iomove() this is msg */
#define MSGIN  0                                    /* same as B_WRITE */
#define MSGOUT          01                                   /* same as B_READ */

#define MDISAB          0
#define MENAB 1
#define MSEND 2
#define MSENDW          3
#define MRECV 4
#define MRECVW          5
#define MSTAT  6
#define MSGCTL          7

struct msghdr
{               struct msghdr   *mq_forw;
                int             mq_size;
                int             mq_sender;
                int             mq_type;
};
struct msgqhdr
{               struct msghdr   *mq_forw;       /* note same position as in msghdr */
                struct msghdr   *mq_last;
                int             *mq_procp;
                char            mq_flag;
                char            mq_cnt;
                int             mq_meslim;
};

#endif

/* commands for msgctl call here */
#define SETMQLEN 0                                  /*set mes q length command*/


struct mstat {
                unsigned        ms_cnt;
                unsigned        ms_maxm;
};

struct mstruct {
                int             ms_frompid;
                int             ms_type;
};
```

DIAGNOSTICS

The error bit (c-bit) is set for any one of a number of error conditions. An error occurs when enabling messages if no queue is available for use; it is also erroneous to attempt to disable message reception if it is not enabled. When trying to send messages, errors occur because the message is too long, the receiver has not enabled message reception, the type specified is not valid, the receiver has an excessive number of messages outstanding on its queue, or, for conditional sends, the system message buffers are temporarily full. When receiving messages, errors may occur because the process has not enabled message reception, the requested type or size are invalid, or, for conditional receives, a message of the requested type is not on the queue. It is also illegal to set the message limit (via *msgctl*) to a value larger than defined by MAXMXSGDEF in **ipcomm.h**. From C, a −1 return from any function indicates an error.

ASSEMBLER

(msg = 49.; not in assembler)
(size in r0)
sys msg; function; buf; arg3; type

FILES

/usr/include/sys/ipcomm.h