Bell Telephone Laboratories, Incorporated   - 1 -   PA-1C600-01
PROGRAM APPLICATION INSTRUCTION   Section 17 (f)
Issue 1, 1 October 1977
AT&TCo SPCS

MSG(f)   **MSG(f)**

NAME

msg — send and receive messages (USG Version)

SYNOPSIS

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

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;
```

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)

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 two word buffer into which is placed information about the received message as follows:

MSG(f)   **MSG(f)**

```
struct mstruct {
        int        frompid;     /* processid of sender */
        int        mtype;       /* message type */
};
```

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. Both 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.

SEE ALSO

—

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. From C, a −1 return from any function indicates an error.