NAME

msgsnd − XSI message send operation

SYNOPSIS

#include <sys/msg.h>

int msgsnd(int *msqid***, const void ****msgp***, size_t** *msgsz***, int** *msgflg***);**

DESCRIPTION

The *msgsnd*() function operates on XSI message queues (see the Base Definitions volume of IEEE Std 1003.1-2001, Section 3.224, Message Queue). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in *Realtime* .

The *msgsnd*() function shall send a message to the queue associated with the message queue identifier specified by *msqid*.

The application shall ensure that the argument *msgp* points to a user-defined buffer that contains first a field of type **long** specifying the type of the message, and then a data portion that holds the data bytes of the message. The structure below is an example of what this user-defined buffer might look like:

```
struct mymsg {
   long   mtype;      /* Message type. */
   char   mtext[1];   /* Message text. */
}
```

The structure member *mtype* is a non-zero positive type **long** that can be used by the receiving process for message selection.

The structure member *mtext* is any text of length *msgsz* bytes. The argument *msgsz* can range from 0 to a system-imposed maximum.

The argument *msgflg* specifies the action to be taken if one or more of the following is true:

* The number of bytes already on the queue is equal to **msg_qbytes**; see <*sys/msg.h*>.
* The total number of messages on all queues system-wide is equal to the system-imposed limit.

These actions are as follows:

* If (*msgflg* & IPC_NOWAIT) is non-zero, the message shall not be sent and the calling thread shall return immediately.
* If (*msgflg* & IPC_NOWAIT) is 0, the calling thread shall suspend execution until one of the following occurs:
  * The condition responsible for the suspension no longer exists, in which case the message is sent.
  * The message queue identifier *msqid* is removed from the system; when this occurs, *errno* shall be set equal to [EIDRM] and -1 shall be returned.
  * The calling thread receives a signal that is to be caught; in this case the message is not sent and the calling thread resumes execution in the manner prescribed in *sigaction*() .

Upon successful completion, the following actions are taken with respect to the data structure associated with *msqid*; see <*sys/msg.h*>:

* **msg_qnum** shall be incremented by 1.
* **msg_lspid** shall be set equal to the process ID of the calling process.
* **msg_stime** shall be set equal to the current time.

RETURN VALUE

Upon successful completion, *msgsnd*() shall return 0; otherwise, no message shall be sent, *msgsnd*() shall return -1, and *errno* shall be set to indicate the error.

ERRORS

The *msgsnd*() function shall fail if:

EACCES

Operation permission is denied to the calling process; see *XSI Interprocess Communication* .

EAGAIN

The message cannot be sent for one of the reasons cited above and (*msgflg* & IPC_NOWAIT) is non-zero.

EIDRM

The message queue identifier *msqid* is removed from the system.

EINTR

The *msgsnd*() function was interrupted by a signal.

EINVAL

The value of *msqid* is not a valid message queue identifier, or the value of *mtype* is less than 1; or the value of *msgsz* is less than 0 or greater than the system-imposed limit.

The following sections are informative.

EXAMPLES

Sending a Message

The following example sends a message to the queue identified by the *msqid* argument (assuming that value has previously been set). This call specifies that an error should be reported if no message is available. The message size is calculated directly using the *sizeof* operator.

```
#include <sys/msg.h>
...
int result;
int msqid;
struct message {
   long type;
   char text[20];
} msg;


msg.type = 1;
strcpy(msg.text, "This is message 1");
...
result = msgsnd(msqid, (void *) &msg, sizeof(msg.text), IPC_NOWAIT);
```

APPLICATION USAGE

The POSIX Realtime Extension defines alternative interfaces for interprocess communication (IPC). Application developers who need to use IPC should design their applications so that modules using the IPC routines described in *XSI Interprocess Communication* can be easily modified to use the alternative interfaces.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

XSI Interprocess Communication , *Realtime* , *mq_close*() , *mq_getattr*() , *mq_notify*() , *mq_open*() , *mq_receive*() , *mq_send*() , *mq_setattr*() , *mq_unlink*() , *msgctl*() , *msgget*() , *msgrcv*() , *sigaction*() , the Base Definitions volume of IEEE Std 1003.1-2001, *<sys/msg.h>*

COPYRIGHT