

**NAME**

msgrcv, msgsnd – message operations

**SYNOPSIS**

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

```
#include <sys/ipc.h>
```

```
#include <sys/msg.h>
```

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

```
ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp,
               int msgflg);
```

**DESCRIPTION**

The **msgsnd()** and **msgrcv()** system calls are used, respectively, to send messages to, and receive messages from, a message queue. The calling process must have write permission on the message queue in order to send a message, and read permission to receive a message.

The *msgp* argument is a pointer to caller-defined structure of the following general form:

```
struct msgbuf {
    long mtype;    /* message type, must be > 0 */
    char mtext[1]; /* message data */
};
```

The *mtext* field is an array (or other structure) whose size is specified by *msgsz*, a non-negative integer value. Messages of zero length (i.e., no *mtext* field) are permitted. The *mtype* field must have a strictly positive integer value. This value can be used by the receiving process for message selection (see the description of **msgrcv()** below).

**msgsnd()**

The **msgsnd()** system call appends a copy of the message pointed to by *msgp* to the message queue whose identifier is specified by *msqid*.

If sufficient space is available in the queue, **msgsnd()** succeeds immediately. (The queue capacity is defined by the *msg\_bytes* field in the associated data structure for the message queue. During queue creation this field is initialized to **MSGMNB** bytes, but this limit can be modified using **msgctl(2)**.) If insufficient space is available in the queue, then the default behavior of **msgsnd()** is to block until space becomes available. If **IPC\_NOWAIT** is specified in *msgflg*, then the call instead fails with the error **EAGAIN**.

A blocked **msgsnd()** call may also fail if:

- \* the queue is removed, in which case the system call fails with *errno* set to **EIDRM**; or
- \* a signal is caught, in which case the system call fails with *errno* set to **EINTR**; see **signal(7)**. (**msgsnd()** is never automatically restarted after being interrupted by a signal handler, regardless of the setting of the **SA\_RESTART** flag when establishing a signal handler.)

Upon successful completion the message queue data structure is updated as follows:

*msg\_lspid* is set to the process ID of the calling process.

*msg\_qnum* is incremented by 1.

*msg\_stime* is set to the current time.

**msgrcv()**

The **msgrcv()** system call removes a message from the queue specified by *msqid* and places it in the buffer pointed to by *msgp*.

The argument *msgsz* specifies the maximum size in bytes for the member *mtext* of the structure pointed to by the *msgp* argument. If the message text has length greater than *msgsz*, then the behavior depends on whether **MSG\_NOERROR** is specified in *msgflg*. If **MSG\_NOERROR** is specified, then the message

text will be truncated (and the truncated part will be lost); if **MSG\_NOERROR** is not specified, then the message isn't removed from the queue and the system call fails returning `-1` with *errno* set to **E2BIG**.

The argument *msgtyp* specifies the type of message requested as follows:

- \* If *msgtyp* is 0, then the first message in the queue is read.
- \* If *msgtyp* is greater than 0, then the first message in the queue of type *msgtyp* is read, unless **MSG\_EXCEPT** was specified in *msgflg*, in which case the first message in the queue of type not equal to *msgtyp* will be read.
- \* If *msgtyp* is less than 0, then the first message in the queue with the lowest type less than or equal to the absolute value of *msgtyp* will be read.

The *msgflg* argument is a bit mask constructed by ORing together zero or more of the following flags:

#### **IPC\_NOWAIT**

Return immediately if no message of the requested type is in the queue. The system call fails with *errno* set to **ENOMSG**.

#### **MSG\_EXCEPT**

Used with *msgtyp* greater than 0 to read the first message in the queue with message type that differs from *msgtyp*.

#### **MSG\_NOERROR**

To truncate the message text if longer than *msgsz* bytes.

If no message of the requested type is available and **IPC\_NOWAIT** isn't specified in *msgflg*, the calling process is blocked until one of the following conditions occurs:

- \* A message of the desired type is placed in the queue.
- \* The message queue is removed from the system. In this case the system call fails with *errno* set to **EIDRM**.
- \* The calling process catches a signal. In this case the system call fails with *errno* set to **EINTR**. (**msgrev()** is never automatically restarted after being interrupted by a signal handler, regardless of the setting of the **SA\_RESTART** flag when establishing a signal handler.)

Upon successful completion the message queue data structure is updated as follows:

*msg\_lripid* is set to the process ID of the calling process.

*msg\_qnum* is decremented by 1.

*msg\_rtime* is set to the current time.

#### **RETURN VALUE**

On failure both functions return `-1` with *errno* indicating the error, otherwise **msgsnd()** returns 0 and **msgrev()** returns the number of bytes actually copied into the *mtext* array.

#### **ERRORS**

When **msgsnd()** fails, *errno* will be set to one among the following values:

##### **EACCES**

The calling process does not have write permission on the message queue, and does not have the **CAP\_IPC\_OWNER** capability.

##### **EAGAIN**

The message can't be sent due to the *msg\_qbytes* limit for the queue and **IPC\_NOWAIT** was specified in *msgflg*.

##### **EFAULT**

The address pointed to by *msgp* isn't accessible.

##### **EIDRM**

The message queue was removed.

**EINTR**

Sleeping on a full message queue condition, the process caught a signal.

**EINVAL**

Invalid *msqid* value, or non-positive *mtype* value, or invalid *msgsz* value (less than 0 or greater than the system value **MSGMAX**).

**ENOMEM**

The system does not have enough memory to make a copy of the message pointed to by *msgp*.

When **msgrcv()** fails, *errno* will be set to one among the following values:

**E2BIG** The message text length is greater than *msgsz* and **MSG\_NOERROR** isn't specified in *msgflg*.

**EACCES**

The calling process does not have read permission on the message queue, and does not have the **CAP\_IPC\_OWNER** capability.

**EAGAIN**

No message was available in the queue and **IPC\_NOWAIT** was specified in *msgflg*.

**EFAULT**

The address pointed to by *msgp* isn't accessible.

**EIDRM**

While the process was sleeping to receive a message, the message queue was removed.

**EINTR**

While the process was sleeping to receive a message, the process caught a signal; see **signal(7)**.

**EINVAL**

*msqid* was invalid, or *msgsz* was less than 0.

**ENOMSG**

**IPC\_NOWAIT** was specified in *msgflg* and no message of the requested type existed on the message queue.

**CONFORMING TO**

SVr4, POSIX.1-2001.

**NOTES**

The *msgp* argument is declared as *struct msgbuf \** with libc4, libc5, glibc 2.0, glibc 2.1. It is declared as *void \** with glibc 2.2 and later, as required by SUSv2 and SUSv3.

The following limits on message queue resources affect the **msgsnd()** call:

**MSGMAX**

Maximum size for a message text: 8192 bytes (on Linux, this limit can be read and modified via */proc/sys/kernel/msgmax*).

**MSGMNB**

Default maximum size in bytes of a message queue: 16384 bytes (on Linux, this limit can be read and modified via */proc/sys/kernel/msgmnb*). The superuser can increase the size of a message queue beyond **MSGMNB** by a **msgctl(2)** system call.

The implementation has no intrinsic limits for the system wide maximum number of message headers (**MSGTQL**) and for the system wide maximum size in bytes of the message pool (**MSGPOOL**).

**SEE ALSO**

**msgctl(2)**, **msgget(2)**, **capabilities(7)**, **mq\_overview(7)**, **svipc(7)**

**COLOPHON**

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at <http://www.kernel.org/doc/man-pages/>.