NAME

msgget − get a System V message queue identifier

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <sys/msg.h>

int msgget(key_t *key***, int** *msgflg***);**

DESCRIPTION

The **msgget**() system call returns the System V message queue identifier associated with the value of the *key* argument. It may be used either to obtain the identifier of a previously created message queue (when *msgflg* is zero and *key* does not have the value **IPC_PRIVATE**), or to create a new set.

A new message queue is created if *key* has the value **IPC_PRIVATE** or *key* isn't **IPC_PRIVATE**, no message queue with the given key *key* exists, and **IPC_CREAT** is specified in *msgflg*.

If *msgflg* specifies both **IPC_CREAT** and **IPC_EXCL** and a message queue already exists for *key*, then **msgget**() fails with *errno* set to **EEXIST**. (This is analogous to the effect of the combination **O_CREAT | O_EXCL** for **open**(2).)

Upon creation, the least significant bits of the argument *msgflg* define the permissions of the message queue. These permission bits have the same format and semantics as the permissions specified for the *mode* argument of **open**(2). (The execute permissions are not used.)

If a new message queue is created, then its associated data structure *msqid_ds* (see **msgctl**(2)) is initialized as follows:

- *msg_perm.cuid* and *msg_perm.uid* are set to the effective user ID of the calling process.
- *msg_perm.cgid* and *msg_perm.gid* are set to the effective group ID of the calling process.
- The least significant 9 bits of *msg_perm.mode* are set to the least significant 9 bits of *msgflg*.
- *msg_qnum*, *msg_lspid*, *msg_lrpid*, *msg_stime*, and *msg_rtime* are set to 0.
- *msg_ctime* is set to the current time.
- *msg_qbytes* is set to the system limit **MSGMNB**.

If the message queue already exists the permissions are verified, and a check is made to see if it is marked for destruction.

RETURN VALUE

On success, **msgget**() returns the message queue identifier (a nonnegative integer). On failure, −1 is returned, and *errno* is set to indicate the error.

ERRORS

EACCES
A message queue exists for *key*, but the calling process does not have permission to access the queue, and does not have the **CAP_IPC_OWNER** capability in the user namespace that governs its IPC namespace.

EEXIST
IPC_CREAT and **IPC_EXCL** were specified in *msgflg*, but a message queue already exists for *key*.

ENOENT
No message queue exists for *key* and *msgflg* did not specify **IPC_CREAT**.

ENOMEM
A message queue has to be created but the system does not have enough memory for the new data structure.

ENOSPC
A message queue has to be created but the system limit for the maximum number of message queues (**MSGMNI**) would be exceeded.

STANDARDS

POSIX.1-2001, POSIX.1-2008, SVr4.

NOTES

IPC_PRIVATE isn't a flag field but a *key_t* type. If this special value is used for *key*, the system call ignores everything but the least significant 9 bits of *msgflg* and creates a new message queue (on success).

The following is a system limit on message queue resources affecting a **msgget**() call:

MSGMNI
System-wide limit on the number of message queues. Before Linux 3.19, the default value for this limit was calculated using a formula based on available system memory. Since Linux 3.19, the default value is 32,000. On Linux, this limit can be read and modified via */proc/sys/kernel/msgmni*.

Linux notes

Until Linux 2.3.20, Linux would return **EIDRM** for a **msgget**() on a message queue scheduled for deletion.

BUGS

The name choice **IPC_PRIVATE** was perhaps unfortunate, **IPC_NEW** would more clearly show its function.

SEE ALSO

msgctl(2), **msgrcv**(2), **msgsnd**(2), **ftok**(3), **capabilities**(7), **mq_overview**(7), **sysvipc**(7)