PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

mq_open — open a message queue (**REALTIME**)

SYNOPSIS

#include <mqueue.h>

mqd_t mq_open(const char *_name_, int _oflag_, ...);

DESCRIPTION

The *mq_open*() function shall establish the connection between a process and a message queue with a message queue descriptor. It shall create an open message queue description that refers to the message queue, and a message queue descriptor that refers to that open message queue description. The message queue descriptor is used by other functions to refer to that message queue. The *name* argument points to a string naming a message queue. It is unspecified whether the name appears in the file system and is visible to other functions that take pathnames as arguments. The *name* argument conforms to the construction rules for a pathname, except that the interpretation of <slash> characters other than the leading <slash> character in *name* is implementation-defined, and that the length limits for the *name* argument are implementation-defined and need not be the same as the pathname limits {PATH_MAX} and {NAME_MAX}. If *name* begins with the <slash> character, then processes calling *mq_open*() with the same value of *name* shall refer to the same message queue object, as long as that name has not been removed. If *name* does not begin with the <slash> character, the effect is implementation-defined. If the *name* argument is not the name of an existing message queue and creation is not requested, *mq_open*() shall fail and return an error.

A message queue descriptor may be implemented using a file descriptor, in which case applications can open up to at least {OPEN_MAX} file and message queues.

The *oflag* argument requests the desired receive and/or send access to the message queue. The requested access permission to receive messages or send messages shall be granted if the calling process would be granted read or write access, respectively, to an equivalently protected file.

The value of *oflag* is the bitwise-inclusive OR of values from the following list. Applications shall specify exactly one of the first three values (access modes) below in the value of *oflag*:

O_RDONLY Open the message queue for receiving messages. The process can use the returned message queue descriptor with *mq_receive*(), but not *mq_send*(). A message queue may be open multiple times in the same or different processes for receiving messages.

O_WRONLY Open the queue for sending messages. The process can use the returned message queue descriptor with *mq_send*() but not *mq_receive*(). A message queue may be open multiple times in the same or different processes for sending messages.

O_RDWR Open the queue for both receiving and sending messages. The process can use any of the functions allowed for O_RDONLY and O_WRONLY. A message queue may be open multiple times in the same or different processes for sending messages.

Any combination of the remaining flags may be specified in the value of *oflag*:

O_CREAT Create a message queue. It requires two additional arguments: *mode*, which shall be of type **mode_t**, and *attr*, which shall be a pointer to an **mq_attr** structure. If the pathname *name* has already been used to create a message queue that still exists, then this flag shall have no effect, except as noted under O_EXCL. Otherwise, a message queue shall be created without any messages in it. The user ID of the message queue shall be set to the effective user ID of the process. The group ID of the message queue shall be set to the effective group ID of the process; however, if the *name* argument is visible in the file system, the group ID may be set to the group ID of the containing directory. When bits in *mode* other than the file permission bits are specified, the effect is unspecified. If *attr* is NULL, the message queue shall be created with implementation-defined default message queue attributes. If *attr* is non-NULL and the calling process has appropriate privileges on *name*, the message queue *mq_maxmsg* and *mq_msgsize* attributes shall be set to

the values of the corresponding members in the **mq_attr** structure referred to by *attr*. The values of the *mq_flags* and *mq_curmsgs* members of the **mq_attr** structure shall be ignored. If *attr* is non-NULL, but the calling process does not have appropriate privileges on *name*, the *mq_open*() function shall fail and return an error without creating the message queue.

O_EXCL If O_EXCL and O_CREAT are set, *mq_open*() shall fail if the message queue *name* exists. The check for the existence of the message queue and the creation of the message queue if it does not exist shall be atomic with respect to other threads executing *mq_open*() naming the same *name* with O_EXCL and O_CREAT set. If O_EXCL is set and O_CREAT is not set, the result is undefined.

O_NONBLOCK

Determines whether an *mq_send*() or *mq_receive*() waits for resources or messages that are not currently available, or fails with *errno* set to **[EAGAIN]**; see *mq_send*( ) and *mq_receive*( ) for details.

The *mq_open*() function does not add or remove messages from the queue.

RETURN VALUE

Upon successful completion, the function shall return a message queue descriptor; otherwise, the function shall return (**mqd_t**)−1 and set *errno* to indicate the error.

ERRORS

The *mq_open*() function shall fail if:

EACCES

The message queue exists and the permissions specified by *oflag* are denied, or the message queue does not exist and permission to create the message queue is denied.

EEXIST

O_CREAT and O_EXCL are set and the named message queue already exists.

EINTR

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

EINVAL

The *mq_open*() function is not supported for the given name.

EINVAL

O_CREAT was specified in *oflag*, the value of *attr* is not NULL, and either *mq_maxmsg* or *mq_msgsize* was less than or equal to zero.

EMFILE

Too many message queue descriptors or file descriptors are currently in use by this process.

ENFILE

Too many message queues are currently open in the system.

ENOENT

O_CREAT is not set and the named message queue does not exist.

ENOSPC

There is insufficient space for the creation of the new message queue.

If any of the following conditions occur, the *mq_open*() function may return (**mqd_t**)−1 and set *errno* to the corresponding value.

ENAMETOOLONG

The length of the *name* argument exceeds {_POSIX_PATH_MAX} on systems that do not support the XSI option or exceeds {_XOPEN_PATH_MAX} on XSI systems, or has a pathname component that is longer than {_POSIX_NAME_MAX} on systems that do not support the XSI option or longer than {_XOPEN_NAME_MAX} on XSI systems.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

None.

FUTURE DIRECTIONS

A future version might require the *mq_open*() and *mq_unlink*() functions to have semantics similar to normal file system operations.

SEE ALSO

mq_close( ), *mq_getattr*( ), *mq_receive*( ), *mq_send*( ), *mq_setattr*( ), *mq_unlink*( ), *msgctl*( ), *msgget*( ), *msgrcv*( ), *msgsnd*( )

The Base Definitions volume of POSIX.1-2008, **<mqueue.h>**

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2013 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, Copyright (C) 2013 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. (This is POSIX.1-2008 with the 2013 Technical Corrigendum 1 applied.) In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.unix.org/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.kernel.org/doc/man-pages/reporting_bugs.html .