



## ***Rocky Enterprise Linux 9.2 Manual Pages on command 'mq\_open.2'***

**C:\>man mq\_open.2**

MQ\_OPEN(3)                   Linux Programmer's Manual                   MQ\_OPEN(3)

### NAME

mq\_open - open a message queue

### SYNOPSIS

```
#include <fcntl.h>           /* For O_* constants */
#include <sys/stat.h>       /* For mode constants */
#include <mqueue.h>

mqd_t mq_open(const char *name, int oflag);
mqd_t mq_open(const char *name, int oflag, mode_t mode,
              struct mq_attr *attr);
```

Link with -lrt.

### DESCRIPTION

mq\_open() creates a new POSIX message queue or opens an existing queue. The queue is identified by name. For details of the construction of name, see mq\_overview(7).

The oflag argument specifies flags that control the operation of the call. (Definitions of the flags values can be obtained by including <fcntl.h>.) Exactly one of the following must be specified in oflag:

**O\_RDONLY**

Open the queue to receive messages only.

**O\_WRONLY**

Open the queue to send messages only.

O\_RDWR Open the queue to both send and receive messages.

Zero or more of the following flags can additionally be ORed in oflag:

O\_CLOEXEC (since Linux 2.6.26)

Set the close-on-exec flag for the message queue descriptor. See open(2) for a discussion of why this flag is useful.

O\_CREAT

Create the message queue if it does not exist. The owner (user ID) of the message queue is set to the effective user ID of the calling process. The group ownership (group ID) is set to the effective group ID of the calling process.

O\_EXCL If O\_CREAT was specified in oflag, and a queue with the given name already exists, then fail with the error EEXIST.

O\_NONBLOCK

Open the queue in nonblocking mode. In circumstances where mq\_receive(3) and mq\_send(3) would normally block, these functions instead fail with the error EAGAIN.

If O\_CREAT is specified in oflag, then two additional arguments must be supplied.

The mode argument specifies the permissions to be placed on the new queue, as for open(2). (Symbolic definitions for the permissions bits can be obtained by including <sys/stat.h>.)

The permissions settings are masked against the process umask.

The fields of the struct mq\_attr pointed to attr specify the maximum number of messages and the maximum size of messages that the queue will allow. This structure is defined as follows:

```
struct mq_attr {
    long mq_flags;    /* Flags (ignored for mq_open()) */
    long mq_maxmsg;  /* Max. # of messages on queue */
    long mq_msgsize; /* Max. message size (bytes) */
    long mq_curmsgs; /* # of messages currently in queue
                     (ignored for mq_open()) */
};
```

Only the mq\_maxmsg and mq\_msgsize fields are employed when calling mq\_open(); the values in the remaining fields are ignored.

If attr is NULL, then the queue is created with implementation-defined default attr

tributes. Since Linux 3.5, two /proc files can be used to control these defaults; see mq\_overview(7) for details.

## RETURN VALUE

On success, mq\_open() returns a message queue descriptor for use by other message queue functions. On error, mq\_open() returns (mqd\_t) -1, with errno set to indicate the error.

## ERRORS

**EACCES** The queue exists, but the caller does not have permission to open it in the specified mode.

**EACCES** name contained more than one slash.

**EEXIST** Both O\_CREAT and O\_EXCL were specified in oflag, but a queue with this name already exists.

**EINVAL** name doesn't follow the format in mq\_overview(7).

**EINVAL** O\_CREAT was specified in oflag, and attr was not NULL, but attr->mq\_maxmsg or attr->mq\_msgsize was invalid. Both of these fields must be greater than zero. In a process that is unprivileged (does not have the CAP\_SYS\_RESOURCE capability), attr->mq\_maxmsg must be less than or equal to the msg\_max limit, and attr->mq\_msgsize must be less than or equal to the msgsize\_max limit. In addition, even in a privileged process, attr->mq\_maxmsg cannot exceed the HARD\_MAX limit. (See mq\_overview(7) for details of these limits.)

**EMFILE** The per-process limit on the number of open file and message queue descriptors has been reached (see the description of RLIMIT\_NOFILE in getrlimit(2)).

## ENAMETOOLONG

name was too long.

**ENFILE** The system-wide limit on the total number of open files and message queues has been reached.

**ENOENT** The O\_CREAT flag was not specified in oflag, and no queue with this name exists.

**ENOENT** name was just "/" followed by no other characters.

**ENOMEM** Insufficient memory.

**ENOSPC** Insufficient space for the creation of a new message queue. This probably

occurred because the queues\_max limit was encountered; see mq\_overview(7).

## ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

??

?Interface ? Attribute ? Value ?

??

?mq\_open() ? Thread safety ? MT-Safe ?

??

## CONFORMING TO

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

## NOTES

### C library/kernel differences

The mq\_open() library function is implemented on top of a system call of the same name. The library function performs the check that the name starts with a slash (/), giving the EINVAL error if it does not. The kernel system call expects name to contain no preceding slash, so the C library function passes name without the preceding slash (i.e., name+1) to the system call.

## BUGS

In kernels before 2.6.14, the process umask was not applied to the permissions specified in mode.

## SEE ALSO

mq\_close(3), mq\_getattr(3), mq\_notify(3), mq\_receive(3), mq\_send(3), mq\_unlink(3), mq\_overview(7)

## COLOPHON

This page is part of release 5.05 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.