NAME

mq_open – open a message queue

SYNOPSIS

cc [*flag* ...] *file* ... **–lposix4** [*library* ...]

#include <mqueue.h>

mqd_t mq_open(const char **name*, **int** *oflag*,
    /* unsigned long *mode*, **mq_attr** *attr* ***/ ...);**

```
struct      mq_attr {
      long    mq_flags;      /* message queue flags */
      long    mq_maxmsg;     /* maximum number of messages */
      long    mq_msgsize;    /* maximum message size */
      long    mq_curmsgs;    /* number of messages currently queued */
      ...
};
```

MT-LEVEL

MT-Safe

DESCRIPTION

mq_open( ) establishes a connection to a named message queue, *name,* returning the address of the message queue descriptor to the caller for subsequent calls to **mq_send**(3R) or **mq_receive**(3R). The message queue once opened remains usable by this process until the message queue is closed by a successful call to **mq_close**(3R), **exit**(2), or **exec**(2).

name points to a string naming a message queue. The *name* argument must conform to the construction rules for a path-name. If *name* is not the name of an existing message queue and its creation is not requested, **mq_open( )** fails and returns an error.

oflag requests the desired receive and/or send access to the message queue. The requested access permission to receive messages or send messages is granted if the calling process would be granted read or write access, respectively, to a file with the equivalent permissions.

The value of *oflag* is the bitwise inclusive OR of values from the following list. Applications must 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**(3R), but not **mq_send**(3R). 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**(3R) but not **mq_receive**(3R). 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 additionally be specified in the value of *oflag*:

O_CREAT This option is used to create a message queue, and it requires two additional arguments: *mode*, which is of type **mode_t**, and *attr*, which is pointer to a **mq_attr** structure. If the pathname, *name*, has already been used to create a message queue that still exists, then this flag has no effect, unless combined with **O_EXCL** (see below). Otherwise, a message queue is created without any messages in it.

The message queue's user ID is set to the process's effective user ID, and the message queue's group ID is set to the process's effective group ID. The message queue's permission bits will be set to the value of *mode,* and modified by clearing all bits set in the file mode creation mask of the process (see **umask**(2)). "AND-NOT") those already set in the file mode creation mask of the process.

If *attr* is **NULL,** the message queue is created with the default message queue attributes, (**mq_maxmsg** = 128 and **mq_maxsize** = 1024). If *attr* is non-**NULL**, the message queue **mq_maxmsg** and **mq_msgsize** attributes are set to the values of the corresponding members in the **mq_attr** structure referred to by *attr*.

O_EXCL If both **O_EXCL** and **O_CREAT** are set, **mq_open( )** will 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 are atomic with respect to other processes executing **mq_open( )** naming the same *name* with both **O_EXCL** and **O_CREAT** set.

O_NONBLOCK The setting of this flag is associated with the open message queue descriptor and determines whether a calling **mq_send**(3R) waits for message buffer space or a calling **mq_receive**(3R) waits for messages that are not currently available; or whether the calling function fails, thereby setting **errno** to **EAGAIN.**

RETURN VALUES

Upon successful completion, **mq_open( )** returns a message queue descriptor; otherwise the function returns **(mqd_t)(-1)** and sets **errno** to indicate the error condition.

ERRORS

EACCESS 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( )** operation was interrupted by a signal.

EINVAL *name* is not a valid name.

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 The number of open message queue descriptors in this process exceeds **{MQ_OPEN_MAX}.**

The number of open file descriptors in this process exceeds **{OPEN_MAX}.**

ENAMETOOLONG
The length of the *name* string exceeds **PATH_MAX,** or a pathname component is longer than **NAME_MAX** while **_POSIX_NO_TRUNC** is in effect.

ENFILE The system file table is full

ENOENT **O_CREAT** is not set and the named message queue, *name*, does not exist.

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

ENOSYS **mq_open( )** is not supported by this implementation.

SEE ALSO

exec(2), **exit**(2), **umask**(2), **sysconf**(3C), **mq_close**(3R), **mq_receive**(3R), **mq_send**(3R), **mq_setattr**(3R), **mq_unlink**(3R),

NOTES

In Solaris, message queues are based on shared memory. Although permissions to send and receive messages are checked by the **mq_receive**() and **mq_send**() interfaces, any application which can open the message queue can directly access the shared memory to examine and manipulate messages in the queue. Thus message queues should not be considered secure.

BUGS

In Solaris 2.5, these functions always return −**1** and set **errno** to **ENOSYS,** because this release does not support the Message Passing option. It is our intention to provide support for these interfaces in future releases.