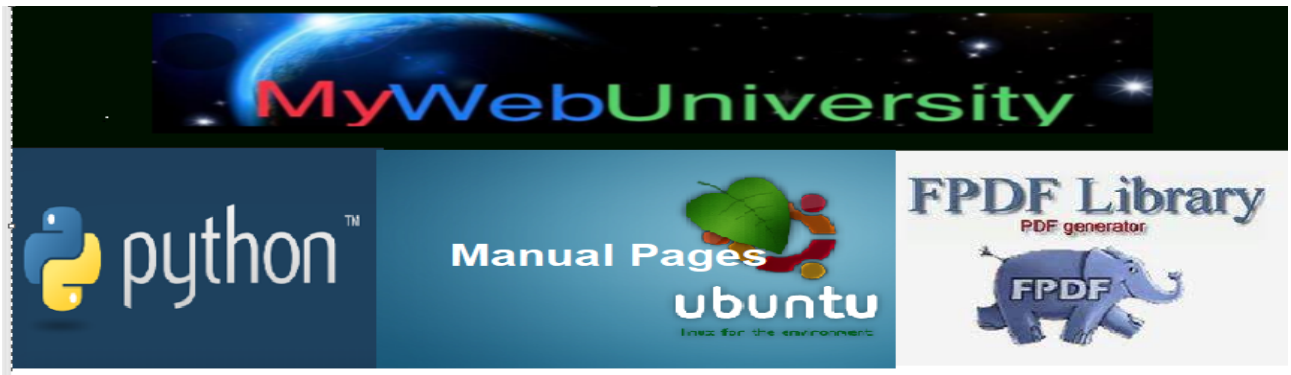

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'mq_open.2'

$ man mq_open.2

mq_open(3) Library Functions Manual mq_open(3)

NAME

mq_open - open a message queue

LIBRARY

Real-time library (librt, -lrt)

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);
```

DESCRIPTION

mq_open() creates a new POSIX message queue or opens an existing queue. The queue is iden? tified 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 dis? cussion 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 per? missions 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 fol? lows:

```
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 attributes. 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_msqsize 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 privi? leged 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 ?

??

VERSIONS

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.

STANDARDS

POSIX.1-2008.

HISTORY

POSIX.1-2001.

BUGS

Before Linux 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)