Rocky Enterprise Linux 9.2 Manual Pages on command 'mq_overview.7'

$ man mq_overview.7

MQ_OVERVIEW(7) Linux Programmer's Manual MQ_OVERVIEW(7)

NAME

mq_overview - overview of POSIX message queues

DESCRIPTION

POSIX message queues allow processes to exchange data in the form of messages. This API is distinct from that provided by System V message queues (msgget(2), msgsnd(2), msgrcv(2), etc.), but provides similar functionality.

Message queues are created and opened using mq_open(3); this function returns a message queue descriptor (mqd_t), which is used to refer to the open message queue in later calls. Each message queue is identi? fied by a name of the form /somename; that is, a null-terminated string of up to NAME_MAX (i.e., 255) characters consisting of an initial slash, followed by one or more characters, none of which are slashes.

Two processes can operate on the same queue by passing the same name to mq_open(3).

Messages are transferred to and from a queue using mq_send(3) and mq_receive(3). When a process has finished using the queue, it closes

it using mq_close(3), and when the queue is no longer required, it can be deleted using mq_unlink(3). Queue attributes can be retrieved and (in some cases) modified using mq_getattr(3) and mq_setattr(3). A process can request asynchronous notification of the arrival of a mes? sage on a previously empty queue using mq_notify(3).

A message queue descriptor is a reference to an open message queue de? scription (see open(2)). After a fork(2), a child inherits copies of its parent's message queue descriptors, and these descriptors refer to the same open message queue descriptions as the corresponding message queue descriptors in the parent. Corresponding message queue descrip? tors in the two processes share the flags (mq_flags) that are associ? ated with the open message queue description.

Each message has an associated priority, and messages are always deliv? ered to the receiving process highest priority first. Message priori? ties range from 0 (low) to sysconf(_SC_MQ_PRIO_MAX) - 1 (high). On Linux, sysconf(_SC_MQ_PRIO_MAX) returns 32768, but POSIX.1 requires only that an implementation support at least priorities in the range 0 to 31; some implementations provide only this range.

The remainder of this section describes some specific details of the Linux implementation of POSIX message queues.

Library interfaces and system calls

In most cases the mq_*() library interfaces listed above are imple? mented on top of underlying system calls of the same name. Deviations from this scheme are indicated in the following table:

Library interface	System call
mq_close(3)	close(2)
mq_getattr(3)	mq_getsetattr(2)
mq_notify(3)	mq_notify(2)
mq_open(3)	mq_open(2)
mq_receive(3)	mq_timedreceive(2)
mq_send(3)	mq_timedsend(2)
mq_setattr(3)	mq_getsetattr(2)
mq_timedreceive(3)	mq_timedreceive(2)

mq_timedsend(3) mq_timedsend(2)

mq_unlink(3) mq_unlink(2)

Versions

POSIX message queues have been supported on Linux since kernel 2.6.6.

Glibc support has been provided since version 2.3.4.

Kernel configuration

Support for POSIX message queues is configurable via the CON?

FIG_POSIX_MQUEUE kernel configuration option. This option is enabled

by default.

Persistence

POSIX message queues have kernel persistence: if not removed by mq_un?

link(3), a message queue will exist until the system is shut down.

Linking

Programs using the POSIX message queue API must be compiled with cc

-lrt to link against the real-time library, librt.

/proc interfaces

The following interfaces can be used to limit the amount of kernel mem?

ory consumed by POSIX message queues and to set the default attributes

for new message queues:

/proc/sys/fs/mqueue/msg_default (since Linux 3.5)

This file defines the value used for a new queue's mq_maxmsg

setting when the queue is created with a call to mq_open(3)

where attr is specified as NULL. The default value for this

file is 10. The minimum and maximum are as for

/proc/sys/fs/mqueue/msg_max. A new queue's default mq_maxmsg

value will be the smaller of msg_default and msg_max. Up until

Linux 2.6.28, the default mq_maxmsg was 10; from Linux 2.6.28 to

Linux 3.4, the default was the value defined for the msg_max

limit.

/proc/sys/fs/mqueue/msg_max

This file can be used to view and change the ceiling value for

the maximum number of messages in a queue. This value acts as a

ceiling on the attr->mq_maxmsg argument given to mq_open(3).

The default value for msg_max is 10. The minimum value is 1 (10 in kernels before 2.6.28). The upper limit is HARD_MSGMAX. The msg_max limit is ignored for privileged processes (CAP_SYS_RE? SOURCE), but the HARD_MSGMAX ceiling is nevertheless imposed. The definition of HARD_MSGMAX has changed across kernel ver? sions:

* Up to Linux 2.6.32: 131072 / sizeof(void *)
* Linux 2.6.33 to 3.4: (32768 * sizeof(void *) / 4)
* Since Linux 3.5: 65,536

/proc/sys/fs/mqueue/msgsize_default (since Linux 3.5)

This file defines the value used for a new queue's mq_msgsize setting when the queue is created with a call to mq_open(3) where attr is specified as NULL. The default value for this file is 8192 (bytes). The minimum and maximum are as for /proc/sys/fs/mqueue/msgsize_max. If msgsize_default exceeds ms? gsize_max, a new queue's default mq_msgsize value is capped to the msgsize_max limit. Up until Linux 2.6.28, the default mq_msgsize was 8192; from Linux 2.6.28 to Linux 3.4, the default was the value defined for the msgsize_max limit.

/proc/sys/fs/mqueue/msgsize_max

This file can be used to view and change the ceiling on the max? imum message size. This value acts as a ceiling on the attr->mq_msgsize argument given to mq_open(3). The default value for msgsize_max is 8192 bytes. The minimum value is 128 (8192 in kernels before 2.6.28). The upper limit for msg? size_max has varied across kernel versions:

* Before Linux 2.6.28, the upper limit is INT_MAX.
* From Linux 2.6.28 to 3.4, the limit is 1,048,576.
* Since Linux 3.5, the limit is 16,777,216 (HARD_MSGSIZEMAX).

The msgsize_max limit is ignored for privileged process (CAP_SYS_RESOURCE), but, since Linux 3.5, the HARD_MSGSIZEMAX ceiling is enforced for privileged processes.

/proc/sys/fs/mqueue/queues_max

This file can be used to view and change the system-wide limit on the number of message queues that can be created. The de? fault value for queues_max is 256. No ceiling is imposed on the queues_max limit; privileged processes (CAP_SYS_RESOURCE) can exceed the limit (but see BUGS).

Resource limit

The RLIMIT_MSGQUEUE resource limit, which places a limit on the amount of space that can be consumed by all of the message queues belonging to a process's real user ID, is described in getrlimit(2).

Mounting the message queue filesystem

On Linux, message queues are created in a virtual filesystem. (Other implementations may also provide such a feature, but the details are likely to differ.) This filesystem can be mounted (by the superuser) using the following commands:

```
# mkdir /dev/mqueue
# mount -t mqueue none /dev/mqueue
```

The sticky bit is automatically enabled on the mount directory.

After the filesystem has been mounted, the message queues on the system can be viewed and manipulated using the commands usually used for files (e.g., ls(1) and rm(1)).

The contents of each file in the directory consist of a single line containing information about the queue:

```
$ cat /dev/mqueue/mymq
QSIZE:129     NOTIFY:2    SIGNO:0    NOTIFY_PID:8260
```

These fields are as follows:

QSIZE Number of bytes of data in all messages in the queue (but see BUGS).

NOTIFY_PID

If this is nonzero, then the process with this PID has used mq_notify(3) to register for asynchronous message notification, and the remaining fields describe how notification occurs.

NOTIFY Notification method: 0 is SIGEV_SIGNAL; 1 is SIGEV_NONE; and 2 is SIGEV_THREAD.

SIGNO Signal number to be used for SIGEV_SIGNAL.

Linux implementation of message queue descriptors

On Linux, a message queue descriptor is actually a file descriptor. (POSIX does not require such an implementation.) This means that a message queue descriptor can be monitored using select(2), poll(2), or epoll(7). This is not portable.

The close-on-exec flag (see open(2)) is automatically set on the file descriptor returned by mq_open(2).

IPC namespaces

For a discussion of the interaction of POSIX message queue objects and IPC namespaces, see ipc_namespaces(7).

NOTES

System V message queues (msgget(2), msgsnd(2), msgrcv(2), etc.) are an older API for exchanging messages between processes. POSIX message queues provide a better designed interface than System V message queues; on the other hand POSIX message queues are less widely avail? able (especially on older systems) than System V message queues.

Linux does not currently (2.6.26) support the use of access control lists (ACLs) for POSIX message queues.

BUGS

In Linux versions 3.5 to 3.14, the kernel imposed a ceiling of 1024 (HARD_QUEUESMAX) on the value to which the queues_max limit could be raised, and the ceiling was enforced even for privileged processes. This ceiling value was removed in Linux 3.14, and patches to stable kernels 3.5.x to 3.13.x also removed the ceiling.

As originally implemented (and documented), the QSIZE field displayed the total number of (user-supplied) bytes in all messages in the mes? sage queue. Some changes in Linux 3.5 inadvertently changed the behav? ior, so that this field also included a count of kernel overhead bytes used to store the messages in the queue. This behavioral regression was rectified in Linux 4.2 (and earlier stable kernel series), so that the count once more included just the bytes of user data in messages in the queue.

EXAMPLES

An example of the use of various message queue functions is shown in mq_notify(3).

SEE ALSO

getrlimit(2), mq_getsetattr(2), poll(2), select(2), mq_close(3), mq_getattr(3), mq_notify(3), mq_open(3), mq_receive(3), mq_send(3), mq_unlink(3), epoll(7), namespaces(7)

COLOPHON

This page is part of release 5.10 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/.