



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

\$ man msgctl.2

msgctl(2) System Calls Manual msgctl(2)

NAME

msgctl - System V message control operations

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <sys/msg.h>

int msgctl(int msqid, int op, struct msqid_ds *buf);
```

DESCRIPTION

msgctl() performs the control operation specified by op on the System V message queue with identifier msqid.

The msqid_ds data structure is defined in <sys/msg.h> as follows:

```
struct msqid_ds {
    struct ipc_perm msg_perm; /* Ownership and permissions */
    time_t      msg_stime; /* Time of last msgsnd(2) */
    time_t      msg_rtime; /* Time of last msgrcv(2) */
    time_t      msg_ctime; /* Time of creation or last
                           modification by msgctl() */
    unsigned long msg_cbytes; /* # of bytes in queue */
    msgqnum_t    msg_qnum; /* # number of messages in queue */
    msglen_t     msg_qbytes; /* Maximum # of bytes in queue */
    pid_t        msg_lspid; /* PID of last msgsnd(2) */
    pid_t        msg_lrpid; /* PID of last msgrcv(2) */
};
```

```
};
```

The fields of the `msgqid_ds` structure are as follows:

`msg_perm` This is an `ipc_perm` structure (see below) that specifies the access permissions on the message queue.

`msg_stime` Time of the last `msgsnd(2)` system call.

`msg_rtime` Time of the last `msgrcv(2)` system call.

`msg_ctime` Time of creation of queue or time of last `msgctl()` `IPC_SET` operation.

`msg_cbytes` Number of bytes in all messages currently on the message queue. This is a non-standard Linux extension that is not specified in POSIX.

`msg_qnum` Number of messages currently on the message queue.

`msg_qbytes` Maximum number of bytes of message text allowed on the message queue.

`msg_lspid` ID of the process that performed the last `msgsnd(2)` system call.

`msg_lrpid` ID of the process that performed the last `msgrcv(2)` system call.

The `ipc_perm` structure is defined as follows (the highlighted fields are settable using `IPC_SET`):

```
struct ipc_perm {  
    key_t    __key;    /* Key supplied to msgget(2) */  
    uid_t    uid;      /* Effective UID of owner */  
    gid_t    gid;      /* Effective GID of owner */  
    uid_t    cuid;     /* Effective UID of creator */  
    gid_t    cgid;     /* Effective GID of creator */  
    unsigned short mode; /* Permissions */  
    unsigned short __seq; /* Sequence number */  
};
```

The least significant 9 bits of the mode field of the `ipc_perm` structure define the access permissions for the message queue. The permission bits are as follows:

0400 Read by user

0200 Write by user

0040 Read by group

0020 Write by group

0004 Read by others

0002 Write by others

Bits 0100, 0010, and 0001 (the execute bits) are unused by the system.

Valid values for op are:

IPC_STAT

Copy information from the kernel data structure associated with msqid into the msqid_ds structure pointed to by buf. The caller must have read permission on the message queue.

IPC_SET

Write the values of some members of the msqid_ds structure pointed to by buf to the kernel data structure associated with this message queue, updating also its msg_ctime member.

The following members of the structure are updated: msg_qbytes, msg_perm.uid, msg_perm.gid, and (the least significant 9 bits of) msg_perm.mode.

The effective UID of the calling process must match the owner (msg_perm.uid) or creator (msg_perm.cuid) of the message queue, or the caller must be privileged. Appropriate privilege (Linux: the CAP_SYS_RESOURCE capability) is required to raise the msg_qbytes value beyond the system parameter MSGMNB.

IPC_RMID

Immediately remove the message queue, awakening all waiting reader and writer processes (with an error return and errno set to EIDRM). The calling process must have appropriate privileges or its effective user ID must be either that of the creator or owner of the message queue. The third argument to msgctl() is ignored in this case.

IPC_INFO (Linux-specific)

Return information about system-wide message queue limits and parameters in the structure pointed to by buf. This structure is of type msginfo (thus, a cast is required), defined in <sys/msg.h> if the _GNU_SOURCE feature test macro is defined:

```
struct msginfo {
    int msgpool; /* Size in kibibytes of buffer pool
                  used to hold message data;
                  unused within kernel */
    int msgmap; /* Maximum number of entries in message
                  map; unused within kernel */
    int msgmax; /* Maximum number of bytes that can be
                  written in a single message */
```

```

int msgmnb; /* Maximum number of bytes that can be
            written to queue; used to initialize
            msg_qbytes during queue creation
            (msgget(2)) */

int msgmni; /* Maximum number of message queues */

int msgssz; /* Message segment size;
            unused within kernel */

int msgtql; /* Maximum number of messages on all queues
            in system; unused within kernel */

unsigned short msgseg;
            /* Maximum number of segments;
            unused within kernel */

};

```

The `msgmni`, `msgmax`, and `msgmnb` settings can be changed via `/proc` files of the same name; see `proc(5)` for details.

MSG_INFO (Linux-specific)

Return a `msginfo` structure containing the same information as for `IPC_INFO`, except that the following fields are returned with information about system resources consumed by message queues: the `msgpool` field returns the number of message queues that currently exist on the system; the `msgmap` field returns the total number of messages in all queues on the system; and the `msgtql` field returns the total number of bytes in all messages in all queues on the system.

MSG_STAT (Linux-specific)

Return a `msqid_ds` structure as for `IPC_STAT`. However, the `msqid` argument is not a queue identifier, but instead an index into the kernel's internal array that maintains information about all message queues on the system.

MSG_STAT_ANY (Linux-specific, since Linux 4.17)

Return a `msqid_ds` structure as for `MSG_STAT`. However, `msg_perm.mode` is not checked for read access for `msqid` meaning that any user can employ this operation (just as any user may read `/proc/sysvipc/msg` to obtain the same information).

RETURN VALUE

On success, `IPC_STAT`, `IPC_SET`, and `IPC_RMID` return 0. A successful `IPC_INFO` or `MSG_INFO` operation returns the index of the highest used entry in the kernel's internal array recording

information about all message queues. (This information can be used with repeated MSG_STAT or MSG_STAT_ANY operations to obtain information about all queues on the system.) A successful MSG_STAT or MSG_STAT_ANY operation returns the identifier of the queue whose index was given in msqid.

On failure, -1 is returned and errno is set to indicate the error.

ERRORS

EACCES The argument op is equal to IPC_STAT or MSG_STAT, but the calling process does not have read permission on the message queue msqid, and does not have the CAP_IPC_OWNER capability in the user namespace that governs its IPC namespace.

EFAULT The argument op has the value IPC_SET or IPC_RMID, but the address pointed to by buf isn't accessible.

EIDRM The message queue was removed.

EINVAL Invalid value for op or msqid. Or: for a MSG_STAT operation, the index value specified in msqid referred to an array slot that is currently unused.

EPERM The argument op has the value IPC_SET or IPC_RMID, but the effective user ID of the calling process is not the creator (as found in msg_perm.cuid) or the owner (as found in msg_perm.uid) of the message queue, and the caller is not privileged (Linux: does not have the CAP_SYS_ADMIN capability).

EPERM An attempt (IPC_SET) was made to increase msg_qbytes beyond the system parameter MSG_MAX_B, but the caller is not privileged (Linux: does not have the CAP_SYS_RESOURCE capability).

STANDARDS

POSIX.1-2024 XSI.

HISTORY

SVr4, SUSv1, POSIX.1-2001 XSI.

Various fields in the struct msqid_ds were typed as short under Linux 2.2 and have become long under Linux 2.4. To take advantage of this, a recompilation under glibc-2.1.91 or later should suffice. (The kernel distinguishes old and new calls by an IPC_64 flag in op.)

NOTES

The IPC_INFO, MSG_STAT, and MSG_INFO operations are used by the ipcs(1) program to provide information on allocated resources. In the future these may be modified or moved to a /proc filesystem interface.

SEE ALSO

msgget(2), msgrcv(2), msgsnd(2), capabilities(7), mq_overview(7), sysvipc(7)

Linux man-pages 6.17

2026-02-11

msgctl(2)