The *cmd* argument specifies the command to be performed, on the queue specified by *msqid*.

`IPC_STAT` Fetch the `msqid_ds` structure for this queue, storing it in the structure pointed to by *buf*.

`IPC_SET` Set the following four fields from the structure pointed to by *buf* in the structure associated with this queue: `msg_perm.uid`, `msg_perm.gid`, `msg_perm.mode`, and `msg_qbytes`. This command can be executed only by a process whose effective user ID equals `msg_perm.cuid` or `msg_perm.uid`, or by a process with superuser privileges. Only the superuser can increase the value of `msg_qbytes`.

`IPC_RMID` Remove the message queue from the system and any data still on the queue. This removal is immediate. Any other process still using the message queue will get an error of `EIDRM` on its next attempted operation on the queue. This command can be executed only by a process whose effective user ID equals `msg_perm.cuid` or `msg_perm.uid`, or by a process with superuser privileges.

We'll see that these three commands (`IPC_STAT`, `IPC_SET`, and `IPC_RMID`) are also provided for semaphores and shared memory.

Data is placed onto a message queue by calling `msgsnd`.

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

int msgsnd(int msqid, const void *ptr, size_t nbytes, int flag);
```

Returns: 0 if OK, −1 on error

As we mentioned earlier, each message is composed of a positive long integer type field, a nonnegative length (*nbytes*), and the actual data bytes (corresponding to the length). Messages are always placed at the end of the queue.

ptr points to a long integer that contains the positive integer message type, and it is immediately followed by the message data. (There is no message data if *nbytes* is 0.) If the largest message we send is 512 bytes, we can define the following structure

```
struct mymesg {
  long  mtype;      /* positive message type */
  char  mtext[512]; /* message data, of length nbytes */
};
```

The *ptr* argument is then a pointer to a `mymesg` structure. The message type can be used by the receiver to fetch messages in an order other than first-in, first-out.

A *flag* value of `IPC_NOWAIT` can be specified. This is similar to the nonblocking I/O flag for file I/O (Section 12.2). If the message queue is full (either the total number