The `ioctl` function was included in the Single UNIX Specification only as an extension for dealing with STREAMS devices [Rago 1993], but it was moved to obsolescent status in SUSv4. UNIX System implementations use `ioctl` for many miscellaneous device operations. Some implementations have even extended it for use with regular files.

The prototype that we show corresponds to POSIX.1. FreeBSD 8.0 and Mac OS X 10.6.8 declare the second argument as an `unsigned long`. This detail doesn't matter, since the second argument is always a `#defined` name from a header.

For the ISO C prototype, an ellipsis is used for the remaining arguments. Normally, however, there is only one more argument, and it's usually a pointer to a variable or a structure.

In this prototype, we show only the headers required for the function itself. Normally, additional device-specific headers are required. For example, the `ioctl` commands for terminal I/O, beyond the basic operations specified by POSIX.1, all require the `<termios.h>` header.

Each device driver can define its own set of `ioctl` commands. The system, however, provides generic `ioctl` commands for different classes of devices. Examples of some of the categories for these generic `ioctl` commands supported in FreeBSD are summarized in Figure 3.15.

Category	Constant names	Header	Number of `ioctl`s
disk labels	`DIOxxx`	`<sys/disklabel.h>`	4
file I/O	`FIOxxx`	`<sys/filio.h>`	14
mag tape I/O	`MTIOxxx`	`<sys/mtio.h>`	11
socket I/O	`SIOxxx`	`<sys/sockio.h>`	73
terminal I/O	`TIOxxx`	`<sys/ttycom.h>`	43

Figure 3.15 Common FreeBSD `ioctl` operations

The mag tape operations allow us to write end-of-file marks on a tape, rewind a tape, space forward over a specified number of files or records, and the like. None of these operations is easily expressed in terms of the other functions in the chapter (`read`, `write`, `lseek`, and so on), so the easiest way to handle these devices has always been to access their operations using `ioctl`.

We use the `ioctl` function in Section 18.12 to fetch and set the size of a terminal's window, and in Section 19.7 when we access the advanced features of pseudo terminals.

3.16 `/dev/fd`

Newer systems provide a directory named `/dev/fd` whose entries are files named `0`, `1`, `2`, and so on. Opening the file `/dev/fd/`*n* is equivalent to duplicating descriptor *n*, assuming that descriptor *n* is open.

The `/dev/fd` feature was developed by Tom Duff and appeared in the 8th Edition of the Research UNIX System. It is supported by all of the systems described in this book: FreeBSD 8.0, Linux 3.2.0, Mac OS X 10.6.8, and Solaris 10. It is not part of POSIX.1.