**IOCTL(2) FreeBSD System Calls Manual IOCTL(2)**

<https://www.freebsd.org/cgi/man.cgi?query=ioctl&sektion=2>

NAME

ioctl -- control device

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/ioctl.h>

int

ioctl(int fd, unsigned long request, ...);

DESCRIPTION

The ioctl() system call manipulates the underlying device parameters of

special files. In particular, many operating characteristics of charac-

ter special files (e.g. terminals) may be controlled with ioctl()

requests. The argument fd must be an open file descriptor.

The third argument to ioctl() is traditionally named char \*argp. Most

uses of ioctl(), however, require the third argument to be a caddr\_t or

an int.

An ioctl() request has encoded in it whether the argument is an ``in''

argument or ``out'' argument, and the size of the argument argp in bytes.

Macros and defines used in specifying an ioctl request are located in the

file <sys/ioctl.h>.

GENERIC IOCTLS

Some generic ioctls are not implemented for all types of file descrip-

tors. These include:

FIONREAD int

Get the number of bytes that are immediately available for read-

ing.

FIONWRITE int

Get the number of bytes in the descriptor's send queue. These

bytes are data which has been written to the descriptor but which

are being held by the kernel for further processing. The nature

of the required processing depends on the underlying device. For

TCP sockets, these bytes have not yet been acknowledged by the

other side of the connection.

FIONSPACE int

Get the free space in the descriptor's send queue. This value is

the size of the send queue minus the number of bytes being held

in the queue. Note: while this value represents the number of

bytes that may be added to the queue, other resource limitations

may cause a write not larger than the send queue's space to be

blocked. One such limitation would be a lack of network buffers

for a write to a network connection.

RETURN VALUES

If an error has occurred, a value of -1 is returned and errno is set to

indicate the error.

ERRORS

The ioctl() system call will fail if:

[EBADF] The fd argument is not a valid descriptor.

[ENOTTY] The fd argument is not associated with a character

special device.

[ENOTTY] The specified request does not apply to the kind of

object that the descriptor fd references.

[EINVAL] The request or argp argument is not valid.

[EFAULT] The argp argument points outside the process's allo-

cated address space.

SEE ALSO

execve(2), fcntl(2), intro(4), tty(4)

======================================================================================================================================================

在学习ioctl 时常常跟 read, write 混淆。其实 ioctl 是用来设置硬件控制寄存器，或者读取硬件状态寄存器的数值之类的。

而read,write 是把数据丢入缓冲区，硬件的驱动从缓冲区读取数据一个个发送或者把接收的数据送入缓冲区。

ioctl(keyFd, FIONREAD, &b)

得到缓冲区里有多少字节要被读取，然后将字节数放入b里面。

接下来就可以用read了。

read(keyFd, &b, sizeof(b))

这两个可以用在按键控制上，先是检测按键是否被按下，如果被按下就放在B里，然后user 在读取按键对应数值。

Listing - Getting the number of bytes in the input buffer.

清单 - 读取串行端口输入缓冲区中的字节数

#include <unistd.h>

#include <termios.h>

int fd;

int bytes;

ioctl(fd, FIONREAD, &bytes);

eg:

#include<stdio.h>

#include<stdlib.h>

#include<sys/ioctl.h>

#include<errno.h>

int kbhit(){

int i;

if(ioctl(0,FIONREAD,&i)<0){

printf("ioctl failed, error=%d\n ",errno);

exit(1);

}

return i;

}

main(){

int i=0;

int c=' ';

system("stty raw -echo" );

printf("enter 'q' to quit \n" );

for(;c!='q';++i){

if(kbhit()){

c=getchar();

printf("\n got %c, on iteration %d",c,i);

}

}

system("stty cooked echo" );

return 0;

}

<http://www.cnblogs.com/huangxingkezhan/archive/2012/12/25/2832655.html>

=========================================================================

HISTORY

The ioctl() function appeared in Version 7 AT&T UNIX.

===============================================================================

**ssize\_t read(int fd,void \* buf ,size\_t count);**

函数说明

read()会把参数fd所指的文件传送count个字节到buf指针所指的内存中。若参数count为0，则read()不会有作用并返回0。返回值为实际读取到的字节数，如果返回0，表示已到达文件尾或是无可读取的数据，此外文件读写位置会随读取到的字节移动。

附加说明

如果顺利read()会返回实际读到的字节数，最好能将返回值与参数count作比较，若返回的字节数比要求读取的字节数少，则有可能读到了文件尾、从管道(pipe)或终端机读取，或者是read()被信号中断了读取动作。当有错误发生时则返回-1，错误代码存入errno中，而文件读写位置则无法预期。

错误代码 EINTR 此调用被信号所中断。

EAGAIN 当使用不可阻断I/O时（O\_NONBLOCK），若无数据可读取则返回此值。

EBADF 参数fd 非有效的文件描述词，或该文件已关闭。】

**ssize\_t write (int fd,const void \* buf,size\_t count);**

.函数说明

write()会把参数buf所指的内存写入count个字节到参数fd所指的文件内。当然，文件读写位置也会随之移动。

.返回值

如果顺利write()会返回实际写入的字节数。当有错误发生时则返回-1，错误代码存入errno中。

.错误代码

EINTR 此调用被信号所中断。

EAGAIN 当使用不可阻断I/O 时（O\_NONBLOCK），若无数据可读取则返回此值。

EBADF 参数fd非有效的文件描述词，或该文件已关闭。