NAME

sockatmark - determine whether the read pointer is at the OOB mark

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/socket.h>

int
sockatmark(*int s*);

DESCRIPTION

To find out if the read pointer is currently pointing at the mark in the data stream, the **sockatmark**() function is provided. If **sockatmark**() returns 1, the next read will return data after the mark. Otherwise (assuming out of band data has arrived), the next read will provide data sent by the client prior to transmission of the out of band signal. The routine used in the remote login process to flush output on receipt of an interrupt or quit signal is shown below. It reads the normal data up to the mark (to discard it), then reads the out-of-band byte.

```
#include <sys/socket.h>
...
oob()
{
        int out = FWRITE, mark;
        char waste[BUFSIZ];

        /* flush local terminal output */
        ioctl(1, TIOCFLUSH, (char *)&out);
        for (;;) {
                if ((mark = sockatmark(rem)) < 0) {
                        perror("sockatmark");
                        break;
                }
                if (mark)
                        break;
                (void) read(rem, waste, sizeof (waste));
        }
        if (recv(rem, &mark, 1, MSG_OOB) < 0) {
                perror("recv");
```

```
                ...
        }
        ...
}
```

RETURN VALUES

Upon successful completion, the **sockatmark**() function returns the value 1 if the read pointer is pointing at the OOB mark, 0 if it is not. Otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **sockatmark**() call fails if:

[EBADF] The *s* argument is not a valid descriptor.

[ENOTTY] The *s* argument is a descriptor for a file, not a socket.

SEE ALSO

recv(2), send(2)

Stuart Sechrest, *An Introductory 4.4BSD Interprocess Communication Tutorial*. https://docs.freebsd.org/44doc/psd/20.ipctut/paper.pdf

Samuel J. Leffler, Robert S. Fabry, William N. Joy, Phil Lapsley, Steve Miller, and Chris Torek, *An Advanced 4.4BSD Interprocess Communication Tutorial*, https://docs.freebsd.org/44doc/psd/21.ipc/paper.pdf.

HISTORY

The **sockatmark**() function was introduced by IEEE Std 1003.1-2001 ("POSIX.1"), to standardize the historical SIOCATMARK ioctl(2). The ENOTTY error instead of the usual ENOTSOCK is to match the historical behavior of SIOCATMARK.