NAME

tee − duplicating pipe content

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#define _GNU_SOURCE /* See feature_test_macros(7) */
#include <fcntl.h>

ssize_t tee(int *fd_in***, int** *fd_out***, size_t** *len***, unsigned int** *flags***);**

DESCRIPTION

tee() duplicates up to *len* bytes of data from the pipe referred to by the file descriptor *fd_in* to the pipe referred to by the file descriptor *fd_out*. It does not consume the data that is duplicated from*fd_in*; therefore, that data can be copied by a subsequent **splice**(2).

flags is a bit mask that is composed by ORing together zero or more of the following values:

SPLICE_F_MOVE	Currently has no effect for **tee**(); see **splice**(2).
SPLICE_F_NONBLOCK	Do not block on I/O; see **splice**(2) for further details.
SPLICE_F_MORE	Currently has no effect for **tee**(), but may be implemented in the future; see **splice**(2).
SPLICE_F_GIFT	Unused for **tee**(); see **vmsplice**(2).

RETURN VALUE

Upon successful completion, **tee**() returns the number of bytes that were duplicated between the input and output. A return value of 0 means that there was no data to transfer, and it would not make sense to block, because there are no writers connected to the write end of the pipe referred to by *fd_in*.

On error, **tee**() returns −1 and *errno* is set to indicate the error.

ERRORS

EAGAIN
SPLICE_F_NONBLOCK was specified in *flags* or one of the file descriptors had been marked as nonblocking (**O_NONBLOCK**), and the operation would block.

EINVAL
fd_in or *fd_out* does not refer to a pipe; or *fd_in* and *fd_out* refer to the same pipe.

ENOMEM
Out of memory.

VERSIONS

The **tee**() system call first appeared in Linux 2.6.17; library support was added in glibc 2.5.

STANDARDS

This system call is Linux-specific.

NOTES

Conceptually, **tee**() copies the data between the two pipes. In reality no real data copying takes place though: under the covers, **tee**() assigns data to the output by merely grabbing a reference to the input.

EXAMPLES

The example below implements a basic **tee**(1) program using the **tee**() system call. Here is an example of its use:

```
$ date | ./a.out out.log | cat
Tue Oct 28 10:06:00 CET 2014
$ cat out.log
Tue Oct 28 10:06:00 CET 2014
```

Program source

```
#define _GNU_SOURCE
#include <errno.h>
#include <fcntl.h>
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int
main(int argc, char *argv[])
{
    int      fd;
    ssize_t  len, slen;

    if (argc != 2) {
        fprintf(stderr, "Usage: %s <file>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd == −1) {
        perror("open");
        exit(EXIT_FAILURE);
    }

    for (;;) {
        /*
         * tee stdin to stdout.
         */
        len = tee(STDIN_FILENO, STDOUT_FILENO,
                  INT_MAX, SPLICE_F_NONBLOCK);
        if (len < 0) {
            if (errno == EAGAIN)
                continue;
            perror("tee");
            exit(EXIT_FAILURE);
        }
        if (len == 0)
            break;

        /*
         * Consume stdin by splicing it to a file.
         */
        while (len > 0) {
            slen = splice(STDIN_FILENO, NULL, fd, NULL,
                          len, SPLICE_F_MOVE);
            if (slen < 0) {
                perror("splice");
                exit(EXIT_FAILURE);
            }
            len −= slen;
        }
```

```
           }

           close(fd);
           exit(EXIT_SUCCESS);
       }
```

SEE ALSO

splice(2), **vmsplice**(2), **pipe**(7)