NAME

pipe, **pipe2** - create descriptor pair for interprocess communication

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <unistd.h>

int
pipe(*int fildes[2]*);

int
pipe2(*int fildes[2]*, *int flags*);

DESCRIPTION

The **pipe**() function creates a *pipe*, which is an object allowing bidirectional data flow, and allocates a pair of file descriptors.

The **pipe2**() system call allows control over the attributes of the file descriptors via the *flags* argument. Values for *flags* are constructed by a bitwise-inclusive OR of flags from the following list, defined in <*fcntl.h*>:

O_CLOEXEC Set the close-on-exec flag for the new file descriptors.

O_NONBLOCK Set the non-blocking flag for the ends of the pipe.

If the *flags* argument is 0, the behavior is identical to a call to **pipe**().

By convention, the first descriptor is normally used as the *read end* of the pipe, and the second is normally the *write end*, so that data written to *fildes[1]* appears on (i.e., can be read from) *fildes[0]*. This allows the output of one program to be sent to another program: the source's standard output is set up to be the write end of the pipe, and the sink's standard input is set up to be the read end of the pipe. The pipe itself persists until all its associated descriptors are closed.

A pipe that has had an end closed is considered *widowed*. Writing on such a pipe causes the writing process to receive a SIGPIPE signal. Widowing a pipe is the only way to deliver end-of-file to a reader: after the reader consumes any buffered data, reading a widowed pipe returns a zero count.

The bidirectional nature of this implementation of pipes is not portable to older systems, so it is

recommended to use the convention for using the endpoints in the traditional manner when using a pipe in one direction.

IMPLEMENTATION NOTES

The **pipe**() function calls the **pipe2**() system call. As a result, system call traces such as those captured by dtrace(1) or ktrace(1) will show calls to **pipe2**().

RETURN VALUES

The **pipe**() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **pipe**() and **pipe2**() system calls will fail if:

[EFAULT] *fildes* argument points to an invalid memory location.

[EMFILE] Too many descriptors are active.

[ENFILE] The system file table is full.

[ENOMEM] Not enough kernel memory to establish a pipe.

The **pipe2**() system call will also fail if:

[EINVAL] The *flags* argument is invalid.

SEE ALSO

sh(1), fork(2), read(2), socketpair(2), write(2)

HISTORY

The **pipe**() function appeared in Version 3 AT&T UNIX.

Bidirectional pipes were first used on AT&T System V Release 4 UNIX.

The **pipe2**() function appeared in FreeBSD 10.0.

The **pipe**() function became a wrapper around **pipe2**() in FreeBSD 11.0.