NAME

pipe - create an interprocess channel

SYNOPSIS

```
#include <unistd.h>

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

DESCRIPTION

The pipe() function creates an I/O mechanism called a pipe and returns two file descriptors, fildes[0] and fildes[1]. The files associated with fildes[0] and fildes[1] are streams and are both opened for reading and writing. The O_NDELAY and O_NONBLOCK flags are cleared.

A read from fildes[0] accesses the data written to fildes[1] on a first-in-first-out (FIFO) basis and a read from fildes[1] accesses the data written to fildes[0] also on a FIFO basis.

The FD_CLOEXEC flag will be clear on both file descriptors.

Upon successful completion pipe() marks for update the st_atime, st_ctime, and st_mtime fields of the pipe.

RETURN VALUES

Upon successful completion, 0 is returned. Otherwise, -1 is returned and errno is set to indicate the error.

ERRORS

The pipe() function will fail if:

EMFILE
There are OPEN_MAX-1 or more file descriptors currently open for this process.

ENFILE
A file table entry could not be allocated.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	Async-Signal-Safe

SEE ALSO

sh(1), fcntl(2), fstat(2), getmsg(2), poll(2), putmsg(2), read(2), write(2), attributes(5), streamio(7I)

NOTES

Since a pipe is bi-directional, there are two separate flows of data. Therefore, the size (st_size) returned by a call to fstat(2) with argument fildes[0] or fildes[1] is the number of bytes available for reading from fildes[0] or fildes[1] respectively. Previously, the size (st_size) returned by a call to fstat() with argument fildes[1] (the write-end) was the number of bytes available for reading from fildes[0] (the read-end).