Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'pipe.3p' command

$ man pipe.3p

PIPE(3P) POSIX Programmer's Manual PIPE(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

pipe ? create an interprocess channel

SYNOPSIS

```
#include <unistd.h>
int pipe(int fildes[2]);
```

DESCRIPTION

The pipe() function shall create a pipe and place two file descriptors, one each into the arguments fildes[0] and fildes[1], that refer to the open file descriptions for the read and write ends of the pipe. The file descriptors shall be allocated as described in Section 2.14, File Descriptor Allocation. The O_NONBLOCK and FD_CLOEXEC flags shall be clear on both file descriptors. (The fcntl() function can be used to set both these flags.)

Data can be written to the file descriptor fildes[1] and read from the file descriptor fildes[0]. A read on the file descriptor fildes[0] shall access data written to the file descriptor fildes[1] on a first-in-first-out basis. It is unspecified whether fildes[0] is also open

for writing and whether fildes[1] is also open for reading.

A process has the pipe open for reading (correspondingly writing) if it has a file descriptor open that refers to the read end, fildes[0] (write end, fildes[1]).

The pipe's user ID shall be set to the effective user ID of the calling process.

The pipe's group ID shall be set to the effective group ID of the call?ing process.

Upon successful completion, pipe() shall mark for update the last data access, last data modification, and last file status change timestamps of the pipe.

RETURN VALUE

Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and errno set to indicate the error, no file descriptors shall be allocated and the contents of fildes shall be left unmodified.

ERRORS

The pipe() function shall fail if:

EMFILE All, or all but one, of the file descriptors available to the process are currently open.

ENFILE The number of simultaneously open files in the system would ex?ceed a system-imposed limit.

The following sections are informative.

EXAMPLES

Using a Pipe to Pass Data Between a Parent Process and a Child Process

The following example demonstrates the use of a pipe to transfer data between a parent process and a child process. Error handling is ex?cluded, but otherwise this code demonstrates good practice when using pipes: after the fork() the two processes close the unused ends of the pipe before they commence transferring data.

```
#include <stdlib.h>
#include <unistd.h>
...
int fildes[2];
```

```
const int BSIZE = 100;
char buf[BSIZE];
ssize_t nbytes;
int status;
status = pipe(fildes);
if (status == -1 ) {
    /* an error occurred */
    ...
}
switch (fork()) {
case -1: /* Handle error */
    break;
case 0:  /* Child - reads from pipe */
    close(fildes[1]);                   /* Write end is unused */
    nbytes = read(fildes[0], buf, BSIZE);   /* Get data from pipe */
    /* At this point, a further read would see end-of-file ... */
    close(fildes[0]);                   /* Finished with pipe */
    exit(EXIT_SUCCESS);
default:  /* Parent - writes to pipe */
    close(fildes[0]);                   /* Read end is unused */
    write(fildes[1], "Hello world\n", 12);  /* Write data on pipe */
    close(fildes[1]);                   /* Child will see EOF */
    exit(EXIT_SUCCESS);
}
```

APPLICATION USAGE

None.

RATIONALE

The wording carefully avoids using the verb ``to open'' in order to avoid any implication of use of open(); see also write().

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.14, File Descriptor Allocation, fcntl(), read(), write()

The Base Definitions volume of POSIX.1?2017, <fcntl.h>, <unistd.h>

COPYRIGHT