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 'pclose.3p' command

$ man pclose.3p

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

pclose ? close a pipe stream to or from a process

SYNOPSIS

```
#include <stdio.h>
int pclose(FILE *stream);
```

DESCRIPTION

The pclose() function shall close a stream that was opened by popen(), wait for the command to terminate, and return the termination status of the process that was running the command language interpreter. However, if a call caused the termination status to be unavailable to pclose(), then pclose() shall return -1 with errno set to [ECHILD] to report this situation. This can happen if the application calls one of the following functions:

* wait()
* waitpid() with a pid argument less than or equal to 0 or equal to the process ID of the command line interpreter
* Any other function not defined in this volume of POSIX.1?2017 that

could do one of the above

In any case, pclose() shall not return before the child process created by popen() has terminated.

If the command language interpreter cannot be executed, the child termination status returned by pclose() shall be as if the command language interpreter terminated using exit(127) or _exit(127).

The pclose() function shall not affect the termination status of any child of the calling process other than the one created by popen() for the associated stream.

If the argument stream to pclose() is not a pointer to a stream created by popen(), the result of pclose() is undefined.

If a thread is canceled during execution of pclose(), the behavior is undefined.

RETURN VALUE

Upon successful return, pclose() shall return the termination status of the command language interpreter. Otherwise, pclose() shall return -1 and set errno to indicate the error.

ERRORS

The pclose() function shall fail if:

ECHILD The status of the child process could not be obtained, as described above.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

None.

RATIONALE

There is a requirement that pclose() not return before the child process terminates. This is intended to disallow implementations that return [EINTR] if a signal is received while waiting. If pclose() returned before the child terminated, there would be no way for the application to discover which child used to be associated with the stream, and it could not do the cleanup itself.

If the stream pointed to by stream was not created by popen(), histori?
cal implementations of pclose() return -1 without setting errno. To
avoid requiring pclose() to set errno in this case, POSIX.1?2008 makes
the behavior unspecified. An application should not use pclose() to
close any stream that was not created by popen().
Some historical implementations of pclose() either block or ignore the
signals SIGINT, SIGQUIT, and SIGHUP while waiting for the child process
to terminate. Since this behavior is not described for the pclose()
function in POSIX.1?2008, such implementations are not conforming.
Also, some historical implementations return [EINTR] if a signal is re?
ceived, even though the child process has not terminated. Such imple?
mentations are also considered non-conforming.
Consider, for example, an application that uses:

```
popen("command", "r")
```

to start command, which is part of the same application. The parent
writes a prompt to its standard output (presumably the terminal) and
then reads from the popen()ed stream. The child reads the response from
the user, does some transformation on the response (pathname expansion,
perhaps) and writes the result to its standard output. The parent
process reads the result from the pipe, does something with it, and
prints another prompt. The cycle repeats. Assuming that both processes
do appropriate buffer flushing, this would be expected to work.
To conform to POSIX.1?2008, pclose() must use waitpid(), or some simi?
lar function, instead of wait().
The code sample below illustrates how the pclose() function might be
implemented on a system conforming to POSIX.1?2008.

```
int pclose(FILE *stream)
{
    int stat;
    pid_t pid;
    pid = <pid for process created for stream by popen()>
    (void) fclose(stream);
    while (waitpid(pid, &stat, 0) == -1) {
```

```
            if (errno != EINTR){
                stat = -1;
                break;
            }
        }
        return(stat);
    }
```

FUTURE DIRECTIONS

None.

SEE ALSO

fork(), popen(), wait()

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