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 'popen.3p' command

$ man popen.3p

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

popen ? initiate pipe streams to or from a process

SYNOPSIS

```
#include <stdio.h>
FILE *popen(const char *command, const char *mode);
```

DESCRIPTION

The popen() function shall execute the command specified by the string command. It shall create a pipe between the calling program and the executed command, and shall return a pointer to a stream that can be used to either read from or write to the pipe.

The environment of the executed command shall be as if a child process were created within the popen() call using the fork() function, and the child invoked the sh utility using the call:

```
execl(shell path, "sh", "-c", command, (char *)0);
```

where shell path is an unspecified pathname for the sh utility.

The popen() function shall ensure that any streams from previous popen() calls that remain open in the parent process are closed in the

new child process.

The mode argument to popen() is a string that specifies I/O mode:

1. If mode is r, when the child process is started, its file descrip?
   tor STDOUT_FILENO shall be the writable end of the pipe, and the
   file descriptor fileno(stream) in the calling process, where stream
   is the stream pointer returned by popen(), shall be the readable
   end of the pipe.
2. If mode is w, when the child process is started its file descriptor
   STDIN_FILENO shall be the readable end of the pipe, and the file
   descriptor fileno(stream) in the calling process, where stream is
   the stream pointer returned by popen(), shall be the writable end
   of the pipe.
3. If mode is any other value, the result is unspecified.

After popen(), both the parent and the child process shall be capable of executing independently before either terminates.

Pipe streams are byte-oriented.

RETURN VALUE

Upon successful completion, popen() shall return a pointer to an open stream that can be used to read or write to the pipe. Otherwise, it shall return a null pointer and may set errno to indicate the error.

ERRORS

The popen() function shall fail if:

EMFILE {STREAM_MAX} streams are currently open in the calling process.

The popen() function may fail if:

EMFILE {FOPEN_MAX} streams are currently open in the calling process.

EINVAL The mode argument is invalid.

The popen() function may also set errno values as described by fork() or pipe().

The following sections are informative.

EXAMPLES

Using popen() to Obtain a List of Files from the ls Utility

The following example demonstrates the use of popen() and pclose() to execute the command ls* in order to obtain a list of files in the cur?

rent directory:

```
#include <stdio.h>
...
FILE *fp;
int status;
char path[PATH_MAX];
fp = popen("ls *", "r");
if (fp == NULL)
    /* Handle error */;
while (fgets(path, PATH_MAX, fp) != NULL)
    printf("%s", path);
status = pclose(fp);
if (status == -1) {
    /* Error reported by pclose() */
    ...
} else {
    /* Use macros described under wait() to inspect `status' in order
       to determine success/failure of command executed by popen() */
    ...
}
```

APPLICATION USAGE

Since open files are shared, a mode r command can be used as an input filter and a mode w command as an output filter.

Buffered reading before opening an input filter may leave the standard input of that filter mispositioned. Similar problems with an output filter may be prevented by careful buffer flushing; for example, with fflush().

A stream opened by popen() should be closed by pclose().

The behavior of popen() is specified for values of mode of r and w. Other modes such as rb and wb might be supported by specific implementations, but these would not be portable features. Note that historical implementations of popen() only check to see if the first character of mode is r. Thus, a mode of robert the robot would be treated as mode

r, and a mode of anything else would be treated as mode w.

If the application calls waitpid() or waitid() with a pid argument greater than 0, and it still has a stream that was called with popen() open, it must ensure that pid does not refer to the process started by popen().

To determine whether or not the environment specified in the Shell and Utilities volume of POSIX.1?2017 is present, use the function call:

```
sysconf(_SC_2_VERSION)
```

(See sysconf()).

RATIONALE

The popen() function should not be used by programs that have set user (or group) ID privileges. The fork() and exec family of functions (ex? cept execlp() and execvp()), should be used instead. This prevents any unforeseen manipulation of the environment of the user that could cause execution of commands not anticipated by the calling program.

If the original and popen()ed processes both intend to read or write or read and write a common file, and either will be using FILE-type C functions (fread(), fwrite(), and so on), the rules for sharing file handles must be observed (see Section 2.5.1, Interaction of File De? scriptors and Standard I/O Streams).

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5, Standard I/O Streams, fork(), pclose(), pipe(), sysconf(), system(), wait(), waitid()

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

The Shell and Utilities volume of POSIX.1?2017, sh

COPYRIGHT

event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker?nel.org/doc/man-pages/reporting_bugs.html .