NAME

fpathconf, pathconf − get configuration values for files

SYNOPSIS

#include <unistd.h>

long fpathconf(int *fd*, **int** *name*);
long pathconf(const char * *path*, **int** *name*);

DESCRIPTION

fpathconf() gets a value for the configuration option *name* for the open file descriptor *fd*.

pathconf() gets a value for configuration option *name* for the filename *path*.

The corresponding macros defined in *<unistd.h>* are minimum values; if an application wants to take advantage of values which may change, a call to **fpathconf**() or **pathconf**() can be made, which may yield more liberal results.

Setting *name* equal to one of the following constants returns the following configuration options:

_PC_LINK_MAX
The maximum number of links to the file. If *fd* or *path* refer to a directory, then the value applies to the whole directory. The corresponding macro is **_POSIX_LINK_MAX**.

_PC_MAX_CANON
The maximum length of a formatted input line, where *fd* or *path* must refer to a terminal. The corresponding macro is **_POSIX_MAX_CANON**.

_PC_MAX_INPUT
The maximum length of an input line, where *fd* or *path* must refer to a terminal. The corresponding macro is **_POSIX_MAX_INPUT**.

_PC_NAME_MAX
The maximum length of a filename in the directory *path* or *fd* that the process is allowed to create. The corresponding macro is **_POSIX_NAME_MAX**.

_PC_PATH_MAX
The maximum length of a relative pathname when *path* or *fd* is the current working directory. The corresponding macro is **_POSIX_PATH_MAX**.

_PC_PIPE_BUF
The maximum number of bytes that can be written atomically to a pipe of FIFO. For **fpathconf**(), *fd* should refer to a pipe or FIFO. For **fpathconf**(), *path* should refer to a FIFO or a directory; in the latter case, the returned value corresponds to FIFOs created in that directory. The corresponding macro is **_POSIX_PIPE_BUF**.

_PC_CHOWN_RESTRICTED
This returns a positive value if the use of **chown**(2) and **fchown**(2) for changing a file's user ID is restricted to a process with appropriate privileges, and changing a file's group ID to a value other than the process's effective group ID or one of its supplementary group IDs is restricted to a process with appropriate privileges. According to POSIX.1, this variable shall always be defined with a value other than −1. The corresponding macro is **_POSIX_CHOWN_RESTRICTED**.

If *fd* or *path* refers to a directory, then the return value applies to all files in that directory.

_PC_NO_TRUNC
This returns nonzero if accessing filenames longer than **_POSIX_NAME_MAX** generates an error. The corresponding macro is **_POSIX_NO_TRUNC**.

_PC_VDISABLE
This returns nonzero if special character processing can be disabled, where *fd* or *path* must refer to a terminal.

RETURN VALUE

The return value of these functions is one of the following:

* On error, −1 is returned and *errno* is set to indicate the cause of the error (for example, **EINVAL**, indicating that *name* is invalid).
* If *name* corresponds to a maximum or minimum limit, and that limit is indeterminate, −1 is returned and *errno* is not changed. (To distinguish an indeterminate limit from an error, set *errno* to zero before the call, and then check whether *errno* is nonzero when −1 is returned.)
* If *name* corresponds to an option, a positive value is returned if the option is supported, and −1 is returned if the option is not supported.
* Otherwise, the current value of the option or limit is returned. This value will not be more restrictive than the corresponding value that was described to the application in *<unistd.h>* or *<limits.h>* when the application was compiled.

ERRORS

EACCES
(**pathconf**()) Search permission is denied for one of the directories in the path prefix of *path*.

EBADF
(**fpathconf**()) *fd* is not a valid file descriptor.

EINVAL
name is invalid.

EINVAL
The implementation does not support an association of *name* with the specified file.

ELOOP
(**pathconf**()) Too many symbolic links were encountered while resolving *path*.

ENAMETOOLONG
(**pathconf**()) *path* is too long.

ENOENT
(**pathconf**()) A component of *path* does not exist, or *path* is an empty string.

ENOTDIR
(**pathconf**()) A component used as a directory in *path* is not in fact a directory.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
fpathconf(), **pathconf**()	Thread safety	MT-Safe

CONFORMING TO

POSIX.1-2001, POSIX.1-2008.

NOTES

Files with name lengths longer than the value returned for *name* equal to **_PC_NAME_MAX** may exist in the given directory.

Some returned values may be huge; they are not suitable for allocating memory.

SEE ALSO

getconf(1), **open**(2), **statfs**(2), **confstr**(3), **sysconf**(3)

COLOPHON

This page is part of release 5.05 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man−pages/.