NAME

pathconf, **lpathconf**, **fpathconf** - get configurable pathname variables

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <unistd.h>

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

long
lpathconf(*const char *path*, *int name*);

long
fpathconf(*int fd*, *int name*);

DESCRIPTION

The **pathconf**(), **lpathconf**() and **fpathconf**() system calls provide a method for applications to determine the current value of a configurable system limit or option variable associated with a pathname or file descriptor.

For **pathconf**() and **lpathconf**(), the *path* argument is the name of a file or directory. For **fpathconf**(), the *fd* argument is an open file descriptor. The *name* argument specifies the system variable to be queried. Symbolic constants for each name value are found in the include file <unistd.h>.

The **lpathconf**() system call is like **pathconf**() except in the case where the named file is a symbolic link, in which case **lpathconf**() returns information about the link, while **pathconf**() returns information about the file the link references.

The available values are as follows:

_PC_LINK_MAX
The maximum file link count.

_PC_MAX_CANON
The maximum number of bytes in terminal canonical input line.

_PC_MAX_INPUT

The minimum maximum number of bytes for which space is available in a terminal input queue.

_PC_NAME_MAX
The maximum number of bytes in a file name.

_PC_PATH_MAX
The maximum number of bytes in a pathname.

_PC_PIPE_BUF
The maximum number of bytes which will be written atomically to a pipe.

_PC_CHOWN_RESTRICTED
Return 1 if appropriate privilege is required for the chown(2) system call, otherwise 0. IEEE Std 1003.1-2001 ("POSIX.1") requires appropriate privilege in all cases, but this behavior was optional in prior editions of the standard.

_PC_NO_TRUNC
Return greater than zero if attempts to use pathname components longer than {NAME_MAX} will result in an [ENAMETOOLONG] error; otherwise, such components will be truncated to {NAME_MAX}. IEEE Std 1003.1-2001 ("POSIX.1") requires the error in all cases, but this behavior was optional in prior editions of the standard, and some non-POSIX-compliant file systems do not support this behavior.

_PC_VDISABLE
Returns the terminal character disabling value.

_PC_ASYNC_IO
Return 1 if asynchronous I/O is supported, otherwise 0.

_PC_PRIO_IO
Returns 1 if prioritised I/O is supported for this file, otherwise 0.

_PC_SYNC_IO
Returns 1 if synchronised I/O is supported for this file, otherwise 0.

_PC_ALLOC_SIZE_MIN
Minimum number of bytes of storage allocated for any portion of a file.

_PC_FILESIZEBITS

Number of bits needed to represent the maximum file size.

_PC_REC_INCR_XFER_SIZE
Recommended increment for file transfer sizes between _PC_REC_MIN_XFER_SIZE and _PC_REC_MAX_XFER_SIZE.

_PC_REC_MAX_XFER_SIZE
Maximum recommended file transfer size.

_PC_REC_MIN_XFER_SIZE
Minimum recommended file transfer size.

_PC_REC_XFER_ALIGN
Recommended file transfer buffer alignment.

_PC_SYMLINK_MAX
Maximum number of bytes in a symbolic link.

_PC_ACL_EXTENDED
Returns 1 if an Access Control List (ACL) can be set on the specified file, otherwise 0.

_PC_ACL_NFS4
Returns 1 if an NFSv4 ACLs can be set on the specified file, otherwise 0.

_PC_ACL_PATH_MAX
Maximum number of ACL entries per file.

_PC_CAP_PRESENT
Returns 1 if a capability state can be set on the specified file, otherwise 0.

_PC_INF_PRESENT
Returns 1 if an information label can be set on the specified file, otherwise 0.

_PC_MAC_PRESENT
Returns 1 if a Mandatory Access Control (MAC) label can be set on the specified file, otherwise 0.

_PC_MIN_HOLE_SIZE
If a file system supports the reporting of holes (see lseek(2)), **pathconf**() and **fpathconf**() return a positive number that represents the minimum hole size returned in bytes. The offsets of holes

returned will be aligned to this same value. A special value of 1 is returned if the file system does not specify the minimum hole size but still reports holes.

_PC_DEALLOC_PRESENT
Return 1 if a file system supports hole-punching (see fspacectl(2)), otherwise 0.

RETURN VALUES

If the call to **pathconf**() or **fpathconf**() is not successful, -1 is returned and *errno* is set appropriately. Otherwise, if the variable is associated with functionality that does not have a limit in the system, -1 is returned and *errno* is not modified. Otherwise, the current variable value is returned.

ERRORS

If any of the following conditions occur, the **pathconf**() and **fpathconf**() system calls shall return -1 and set *errno* to the corresponding value.

[EINVAL] The value of the *name* argument is invalid.

[EINVAL] The implementation does not support an association of the variable name with the associated file.

The **pathconf**() system call will fail if:

[ENOTDIR] A component of the path prefix is not a directory.

[ENAMETOOLONG]
A component of a pathname exceeded {NAME_MAX} characters (but see _PC_NO_TRUNC above), or an entire path name exceeded {PATH_MAX} characters.

[ENOENT] The named file does not exist.

[EACCES] Search permission is denied for a component of the path prefix.

[ELOOP] Too many symbolic links were encountered in translating the pathname.

[EIO] An I/O error occurred while reading from or writing to the file system.

[EINTEGRITY] Corrupted data was detected while reading from the file system.

The **fpathconf**() system call will fail if:

[EBADF] The *fd* argument is not a valid open file descriptor.

[EIO] An I/O error occurred while reading from or writing to the file system.

[EINTEGRITY] Corrupted data was detected while reading from the file system.

SEE ALSO

lseek(2), sysctl(3)

HISTORY

The **pathconf**() and **fpathconf**() system calls first appeared in 4.4BSD. The **lpathconf**() system call first appeared in FreeBSD 8.0.