fpathconf - Man Page

get configurable pathname variables

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.

Synopsis

```
#include <unistd.h>

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

Description

The *fpathconf*() and *pathconf*() functions shall determine the current value of a configurable limit or option (*variable*) that is associated with a file or directory.

For *pathconf*(), the *path* argument points to the pathname of a file or directory.

For *fpathconf*(), the *fildes* argument is an open file descriptor.

The *name* argument represents the variable to be queried relative to that file or directory. Implementations shall support all of the variables listed in the following table and may support others. The variables in the following table come from *<limits.h>* or *<unistd.h>* and

fpathconf - Man Page

Variable	Value of *name*	Requirements
{FILESIZEBITS}	_PC_FILESIZEBITS	4,7
{LINK_MAX}	_PC_LINK_MAX	1
{MAX_CANON}	_PC_MAX_CANON	2
{MAX_INPUT}	_PC_MAX_INPUT	2
{NAME_MAX}	_PC_NAME_MAX	3,4
{PATH_MAX}	_PC_PATH_MAX	4,5
{PIPE_BUF}	_PC_PIPE_BUF	6
{POSIX2_SYMLINKS}	_PC_2_SYMLINKS	4
{POSIX_ALLOC_SIZE_MIN}	_PC_ALLOC_SIZE_MIN	10
{POSIX_REC_INCR_XFER_SIZE}	_PC_REC_INCR_XFER_SIZE	10
{POSIX_REC_MAX_XFER_SIZE}	_PC_REC_MAX_XFER_SIZE	10
{POSIX_REC_MIN_XFER_SIZE}	_PC_REC_MIN_XFER_SIZE	10
{POSIX_REC_XFER_ALIGN}	_PC_REC_XFER_ALIGN	10
{SYMLINK_MAX}	_PC_SYMLINK_MAX	4,9
_POSIX_CHOWN_RESTRICTED	_PC_CHOWN_RESTRICTED	7
_POSIX_NO_TRUNC	_PC_NO_TRUNC	3,4
_POSIX_VDISABLE	_PC_VDISABLE	2
_POSIX_ASYNC_IO	_PC_ASYNC_IO	8
_POSIX_PRIO_IO	_PC_PRIO_IO	8
_POSIX_SYNC_IO	_PC_SYNC_IO	8
_POSIX_TIMESTAMP_RESOLUTION	_PC_TIMESTAMP_RESOLUTION	1

Requirements

2. If *path* or *fildes* does not refer to a terminal file, it is unspecified whether an implementation supports an association of the variable name with the specified file.
3. If *path* or *fildes* refers to a directory, the value returned shall apply to filenames within the directory.
4. If *path* or *fildes* does not refer to a directory, it is unspecified whether an implementation supports an association of the variable name with the specified file.
5. If *path* or *fildes* refers to a directory, the value returned shall be the maximum length of a relative pathname that would not cross any mount points when the specified directory is the working directory.
6. If *path* refers to a FIFO, or *fildes* refers to a pipe or FIFO, the value returned shall apply to the referenced object. If *path* or *fildes* refers to a directory, the value returned shall apply to any FIFO that exists or can be created within the directory. If *path* or *fildes* refers to any other type of file, it is unspecified whether an implementation supports an association of the variable name with the specified file.
7. If *path* or *fildes* refers to a directory, the value returned shall apply to any files, other than directories, that exist or can be created within the directory.
8. If *path* or *fildes* refers to a directory, it is unspecified whether an implementation supports an association of the variable name with the specified file.
9. If *path* or *fildes* refers to a directory, the value returned shall be the maximum length of the string that a symbolic link in that directory can contain.
10. If *path* or *fildes* des does not refer to a regular file, it is unspecified whether an implementation supports an association of the variable name with the specified file. If an implementation

fpathconf - Man Page

Return Value

If *name* is an invalid value, both *pathconf*() and *fpathconf*() shall return −1 and set *errno* to indicate the error.

If the variable corresponding to *name* is described in *<limits.h>* as a maximum or minimum value and the variable has no limit for the path or file descriptor, both *pathconf*() and *fpathconf*() shall return −1 without changing *errno*. Note that indefinite limits do not imply infinite limits; see *<limits.h>*.

If the implementation needs to use *path* to determine the value of *name* and the implementation does not support the association of *name* with the file specified by *path*, or if the process did not have appropriate privileges to query the file specified by *path*, or *path* does not exist, *pathconf*() shall return −1 and set *errno* to indicate the error.

If the implementation needs to use *fildes* to determine the value of *name* and the implementation does not support the association of *name* with the file specified by *fildes*, or if *fildes* is an invalid file descriptor, *fpathconf*() shall return −1 and set *errno* to indicate the error.

Otherwise, *pathconf*() or *fpathconf*() shall return the current variable value for the file or directory without changing *errno*. The value returned shall not be more restrictive than the corresponding value available to the application when it was compiled with the implementation's *<limits.h>* or *<unistd.h>*.

If the variable corresponding to *name* is dependent on an unsupported option, the results are unspecified.

fpathconf - Man Page

The *pathconf*() function shall fail if:

EINVAL

The value of *name* is not valid.

EOVERFLOW

The value of *name* is _PC_TIMESTAMP_RESOLUTION and the resolution is larger than {LONG_MAX}.

The *pathconf*() function may fail if:

EACCES

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

EINVAL

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

ELOOP

A loop exists in symbolic links encountered during resolution of the *path* argument.

ELOOP

More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the *path* argument.

ENAMETOOLONG

The length of a component of a pathname is longer than {NAME_MAX}.

ENAMETOOLONG

with a length that exceeds {PATH_MAX}.

ENOENT

A component of *path* does not name an existing file or *path* is an empty string.

ENOTDIR

A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the *path* argument contains at least one non-<slash> character and ends with one or more trailing <slash> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.

The *fpathconf*() function shall fail if:

EINVAL

The value of *name* is not valid.

EOVERFLOW

The value of *name* is _PC_TIMESTAMP_RESOLUTION and the resolution is larger than {LONG_MAX}.

The *fpathconf*() function may fail if:

EBADF

The *fildes* argument is not a valid file descriptor.

EINVAL

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

fpathconf - Man Page

Examples

None.

Application Usage

Application developers should check whether an option, such as _POSIX_ADVISORY_INFO, is supported prior to obtaining and using values for related variables such as {POSIX_ALLOC_SIZE_MIN}.

Rationale

The *pathconf*() function was proposed immediately after the *sysconf*() function when it was realized that some configurable values may differ across file system, directory, or device boundaries.

For example, {NAME_MAX} frequently changes between System V and BSD-based file systems; System V uses a maximum of 14, BSD 255. On an implementation that provides both types of file systems, an application would be forced to limit all pathname components to 14 bytes, as this would be the value specified in *<limits.h>* on such a system.

Therefore, various useful values can be queried on any pathname or file descriptor, assuming that appropriate privileges are in place.

The value returned for the variable {PATH_MAX} indicates the longest relative pathname that could be given if the specified directory is the current working directory of the process. A process may not always be able to generate a name that long and use it if a subdirectory in the pathname crosses into a more restrictive file system. Note that implementations are allowed to accept pathnames longer than {PATH_MAX} bytes long, but are not allowed to return pathnames longer than this

fpathconf - Man Page

The value returned for the variable _POSIX_CHOWN_RESTRICTED also applies to directories that do not have file systems mounted on them. The value may change when crossing a mount point, so applications that need to know should check for each directory. (An even easier check is to try the *chown*() function and look for an error in case it happens.)

Unlike the values returned by *sysconf*(), the pathname-oriented variables are potentially more volatile and are not guaranteed to remain constant throughout the lifetime of the process. For example, in between two calls to *pathconf*(), the file system in question may have been unmounted and remounted with different characteristics.

Also note that most of the errors are optional. If one of the variables always has the same value on an implementation, the implementation need not look at *path* or *fildes* to return that value and is, therefore, not required to detect any of the errors except the meaning of **[EINVAL]** that indicates that the value of *name* is not valid for that variable, and the **[EOVERFLOW]** error that indicates the value to be returned is larger than {LONG_MAX}.

If the value of any of the limits is unspecified (logically infinite), they will not be defined in *<limits.h>* and the *pathconf*() and *fpathconf*() functions return −1 without changing *errno*. This can be distinguished from the case of giving an unrecognized *name* argument because *errno* is set to **[EINVAL]** in this case.

Since −1 is a valid return value for the *pathconf*() and *fpathconf*() functions, applications should set *errno* to zero before calling them and check *errno* only if the return value is −1.

For the case of {SYMLINK_MAX}, since both *pathconf*() and *open*() follow symbolic links, there is no way that *path* or *fildes* could refer to a symbolic link.

fpathconf - Man Page

    {POSIX_ALLOC_SIZE_MIN} {POSIX_REC_INCR_XFER_SIZE}
    {POSIX_REC_MAX_XFER_SIZE} {POSIX_REC_MIN_XFER_SIZE}
    {POSIX_REC_XFER_ALIGN}

only applied to regular files, but Note 10 also permits implementation of the advisory semantics on other file types unique to an implementation (for example, a character special device).

The **[EOVERFLOW]** error for _PC_TIMESTAMP_RESOLUTION cannot occur on POSIX-compliant file systems because POSIX requires a timestamp resolution no larger than one second. Even on 32-bit systems, this can be represented without overflow.

Future Directions

None.

See Also

chown(), confstr(), sysconf()

The Base Definitions volume of POSIX.1-2017, <limits.h>, <unistd.h>

The Shell and Utilities volume of POSIX.1-2017, getconf

Copyright

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.kernel.org/doc/man-pages/reporting_bugs.html .

Referenced By

chown(3p), confstr(3p), getconf(1p), limits.h(0p), realpath(3p), sysconf(3p), unistd.h(0p).

2017 IEEE/The Open Group POSIX Programmer's Manual

fpathconf - Man Page

Home Blog About