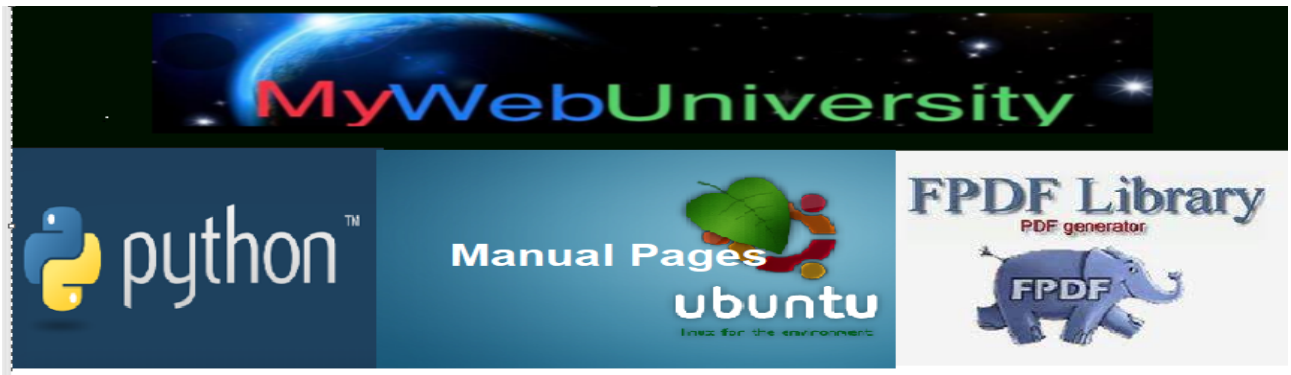


***Full credit is given to the above companies including the OS that this PDF file was generated!***

## *Linux Ubuntu 26.04 Manual Pages on command 'fchdir.2'*

***$ man fchdir.2***

chdir(2) System Calls Manual chdir(2)

NAME

chdir, fchdir - change working directory

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <unistd.h>

int chdir(const char *path);

int fchdir(int fd);
```

Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

fchdir():

_XOPEN_SOURCE >= 500

|| /* Since glibc 2.12: */ _POSIX_C_SOURCE >= 200809L

|| /* glibc up to and including 2.19: */ _BSD_SOURCE

DESCRIPTION

chdir() changes the current working directory of the calling process to the directory speci?

fied in path.

fchdir() is identical to chdir(); the only difference is that the directory is given as an

open file descriptor.

RETURN VALUE

On success, zero is returned. On error, -1 is returned, and errno is set to indicate the

error.

ERRORS

Depending on the filesystem, other errors can be returned. The more general errors for chdir() are listed below:

EACCES Search permission is denied for one of the components of path. (See also path_reso? lution(7).)

EFAULT path points outside your accessible address space.

EIO An I/O error occurred.

ELOOP Too many symbolic links were encountered in resolving path.

ENAMETOOLONG

path is too long.

ENOENT The directory specified in path does not exist.

ENOMEM Insufficient kernel memory was available.

ENOTDIR

A component of path is not a directory.

The general errors for fchdir() are listed below:

EACCES Search permission was denied on the directory open on fd.

EBADF fd is not a valid file descriptor.

ENOTDIR

fd does not refer to a directory.

## STANDARDS

POSIX.1-2024.

## HISTORY

chdir()

SVr4, POSIX.1-1988, 4.4BSD.

fchdir()

SVr4, 4.4BSD, SUSv1, POSIX.1-2001 XSI, POSIX.1-2008.

## NOTES

The current working directory is the starting point for interpreting relative pathnames (those not starting with '/').

A child process created via fork(2) inherits its parent's current working directory. The current working directory is left unchanged by execve(2).

## SEE ALSO

chroot(2), getcwd(3), path_resolution(7)