NAME

faccessat − check user's permissions of a file relative to a directory file descriptor

SYNOPSIS

```
#define _ATFILE_SOURCE
#include <fcntl.h> /* Definition of AT_* constants */
#include <unistd.h>

int faccessat(int dirfd, const char *pathname, int mode, int flags);
```

DESCRIPTION

The **faccessat**() system call operates in exactly the same way as **access**(2), except for the differences described in this manual page.

If the pathname given in *pathname* is relative, then it is interpreted relative to the directory referred to by the file descriptor *dirfd* (rather than relative to the current working directory of the calling process, as is done by **access**(2) for a relative pathname).

If *pathname* is relative and *dirfd* is the special value **AT_FDCWD**, then *pathname* is interpreted relative to the current working directory of the calling process (like **access**(2)).

If *pathname* is absolute, then *dirfd* is ignored.

flags is constructed by ORing together zero or more of the following values:

AT_EACCESS
: Perform access checks using the effective user and group IDs. By default, **faccessat**() uses the real IDs (like **access**(2)).

AT_SYMLINK_NOFOLLOW
: If *pathname* is a symbolic link, do not dereference it: instead return information about the link itself.

RETURN VALUE

On success, (all requested permissions granted) **faccessat**() returns 0. On error, −1 is returned and *errno* is set to indicate the error.

ERRORS

The same errors that occur for **access**(2) can also occur for **faccessat**(). The following additional errors can occur for **faccessat**():

EBADF
: *dirfd* is not a valid file descriptor.

EINVAL
: Invalid flag specified in *flags*.

ENOTDIR
: *pathname* is relative and *dirfd* is a file descriptor referring to a file other than a directory.

VERSIONS

faccessat() was added to Linux in kernel 2.6.16.

CONFORMING TO

POSIX.1-2008.

NOTES

See **openat**(2) for an explanation of the need for **faccessat**().

Glibc Notes

The **AT_EACCESS** and **AT_SYMLINK_NOFOLLOW** flags are actually implemented within the glibc wrapper function for **faccessat**(). If either of these flags are specified, then the wrapper function employs

fstatat(2) to determine access permissions.

SEE ALSO

access(2), **openat**(2), **euidaccess**(3), **credentials**(7), **path_resolution**(7), **symlink**(7)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.