NAME

unlink, **unlinkat**, **funlinkat** - remove directory entry

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <unistd.h>

int
unlink(*const char *path*);

int
unlinkat(*int dfd*, *const char *path*, *int flag*);

int
funlinkat(*int dfd*, *const char *path*, *int fd*, *int flag*);

DESCRIPTION

The **unlink**() system call removes the link named by *path* from its directory and decrements the link count of the file which was referenced by the link. If that decrement reduces the link count of the file to zero, and no process has the file open, then all resources associated with the file are reclaimed. If one or more process have the file open when the last link is removed, the link is removed, but the removal of the file is delayed until all references to it have been closed. The *path* argument may not be a directory.

The **unlinkat**() system call is equivalent to **unlink**() or **rmdir**() except in the case where *path* specifies a relative path. In this case the directory entry to be removed is determined relative to the directory associated with the file descriptor *dfd* instead of the current working directory.

The values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined in <*fcntl.h*>:

AT_REMOVEDIR
: Remove the directory entry specified by *fd* and *path* as a directory, not a normal file.

AT_RESOLVE_BENEATH
: Only walk paths below the directory specified by the *fd* descriptor. See the description of the O_RESOLVE_BENEATH flag in the open(2) manual page.

If **unlinkat**() is passed the special value AT_FDCWD in the *fd* parameter, the current working directory is used and the behavior is identical to a call to *unlink* or *rmdir* respectively, depending on whether or not the AT_REMOVEDIR bit is set in flag.

The **funlinkat**() system call can be used to unlink an already-opened file, unless that file has been replaced since it was opened. It is equivalent to **unlinkat**() in the case where *path* is already open as the file descriptor *fd*. Otherwise, the path will not be removed and an error will be returned. The *fd* can be set the FD_NONE. In that case **funlinkat**() behaves exactly like **unlinkat**().

RETURN VALUES

The **unlink**() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **unlink**() succeeds unless:

[ENOTDIR]	A component of the path prefix is not a directory.
[EISDIR]	The named file is a directory.
[ENAMETOOLONG]	A component of a pathname exceeded 255 characters, or an entire path name exceeded 1023 characters.
[ENOENT]	The named file does not exist.
[EACCES]	Search permission is denied for a component of the path prefix.
[EACCES]	Write permission is denied on the directory containing the link to be removed.
[ELOOP]	Too many symbolic links were encountered in translating the pathname.
[EPERM]	The named file is a directory.
[EPERM]	The named file has its immutable, undeletable or append-only flag set, see the chflags(2) manual page for more information.
[EPERM]	The parent directory of the named file has its immutable or append-only flag set.
[EPERM]	The directory containing the file is marked sticky, and neither the containing directory nor the file to be removed are owned by the effective user ID.

[EIO]	An I/O error occurred while deleting the directory entry or deallocating the inode.
[EINTEGRITY]	Corrupted data was detected while reading from the file system.
[EROFS]	The named file resides on a read-only file system.
[EFAULT]	The *path* argument points outside the process's allocated address space.
[ENOSPC]	On file systems supporting copy-on-write or snapshots, there was not enough free space to record metadata for the delete operation of the file.

In addition to the errors returned by the **unlink**(), the **unlinkat**() may fail if:

[EBADF]	The *path* argument does not specify an absolute path and the *fd* argument is neither AT_FDCWD nor a valid file descriptor open for searching.
[ENOTEMPTY]	The *flag* parameter has the AT_REMOVEDIR bit set and the *path* argument names a directory that is not an empty directory, or there are hard links to the directory other than dot or a single entry in dot-dot.
[ENOTDIR]	The *flag* parameter has the AT_REMOVEDIR bit set and *path* does not name a directory.
[EINVAL]	The value of the *flag* argument is not valid.
[ENOTDIR]	The *path* argument is not an absolute path and *fd* is neither AT_FDCWD nor a file descriptor associated with a directory.
[ENOTCAPABLE]	*path* is an absolute path, or contained a ".." component leading to a directory outside of the directory hierarchy specified by *fd*, and the process is in capability mode or the AT_RESOLVE_BENEATH flag was specified.

In addition to the errors returned by **unlinkat**(), **funlinkat**() may fail if:

[EDEADLK]	The file descriptor is not associated with the path.

SEE ALSO

chflags(2), close(2), link(2), rmdir(2), symlink(7)

STANDARDS

The **unlinkat**() system call follows The Open Group Extended API Set 2 specification.

HISTORY

The **unlink**() function appeared in Version 1 AT&T UNIX. The **unlinkat**() system call appeared in FreeBSD 8.0. The **funlinkat**() system call appeared in FreeBSD 13.0.

The **unlink**() system call traditionally allows the super-user to unlink directories which can damage the file system integrity. This implementation no longer permits it.