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

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'readlink.3p' command

$ man readlink.3p

READLINK(3P) POSIX Programmer's Manual READLINK(3P)

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.

NAME

readlink, readlinkat ? read the contents of a symbolic link

SYNOPSIS

```
#include <unistd.h>
ssize_t readlink(const char *restrict path, char *restrict buf,
    size_t bufsize);
#include <fcntl.h>
ssize_t readlinkat(int fd, const char *restrict path,
    char *restrict buf, size_t bufsize);
```

DESCRIPTION

The readlink() function shall place the contents of the symbolic link referred to by path in the buffer buf which has size bufsize. If the number of bytes in the symbolic link is less than bufsize, the contents of the remainder of buf are unspecified. If the buf argument is not large enough to contain the link content, the first bufsize bytes shall be placed in buf.

If the value of bufsize is greater than {SSIZE_MAX}, the result is im?

plementation-defined.

Upon successful completion, readlink() shall mark for update the last data access timestamp of the symbolic link.

The readlinkat() function shall be equivalent to the readlink() function except in the case where path specifies a relative path. In this case the symbolic link whose content is read is relative to the directory associated with the file descriptor fd instead of the current working directory. If the access mode of the open file description associated with the file descriptor is not O_SEARCH, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the access mode is O_SEARCH, the function shall not perform the check.

If readlinkat() is passed the special value AT_FDCWD in the fd parameter, the current working directory shall be used and the behavior shall be identical to a call to readlink().

RETURN VALUE

Upon successful completion, these functions shall return the count of bytes placed in the buffer. Otherwise, these functions shall return a value of -1, leave the buffer unchanged, and set errno to indicate the error.

ERRORS

These functions shall fail if:

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

EINVAL The path argument names a file that is not a symbolic link.

EIO An I/O error occurred while reading from the file system.

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

ENAMETOOLONG

The length of a component of a pathname is longer than {NAME_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 di? rectory nor a symbolic link to a directory.

The readlinkat() function shall fail if:

EACCES The access mode of the open file description associated with fd is not O_SEARCH and the permissions of the directory underlying fd do not permit directory searches.

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 reading or searching.

ENOTDIR

The path argument is not an absolute path and fd is a file de? scriptor associated with a non-directory file.

These functions may fail if:

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

ENAMETOOLONG

The length of a pathname exceeds {PATH_MAX}, or pathname resolu? tion of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.

The following sections are informative.

EXAMPLES

Reading the Name of a Symbolic Link

The following example shows how to read the name of a symbolic link named /modules/pass1.

```
#include <unistd.h>
char buf[1024];
ssize_t len;
...
```

```
if ((len = readlink("/modules/pass1", buf, sizeof(buf)-1)) != -1)
    buf[len] = '\0';
```

APPLICATION USAGE

Conforming applications should not assume that the returned contents of the symbolic link are null-terminated.

RATIONALE

The type associated with bufsiz is a size_t in order to be consistent with both the ISO C standard and the definition of read(). The behav? ior specified for readlink() when bufsiz is zero represents historical practice. For this case, the standard developers considered a change whereby readlink() would return the number of non-null bytes contained in the symbolic link with the buffer buf remaining unchanged; however, since the stat structure member st_size value can be used to determine the size of buffer necessary to contain the contents of the symbolic link as returned by readlink(), this proposal was rejected, and the historical practice retained.

The purpose of the readlinkat() function is to read the content of sym? bolic links in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to readlink(), resulting in un? specified behavior. By opening a file descriptor for the target direc? tory and using the readlinkat() function it can be guaranteed that the symbolic link read is located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

fstatat(), symlink()

The Base Definitions volume of POSIX.1?2017, <fcntl.h>, <unistd.h>

COPYRIGHT

Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the 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.ker? nel.org/doc/man-pages/reporting_bugs.html .