readlink() — Read the value of a symbolic link

Last Updated: 2021-06-25

Standards

Standards / Extensions	C or C++	Dependencies
POSIX.1a XPG4.2 Single UNIX Specification, Version 3	both	

Format

```
#define _POSIX1_SOURCE 2
#include <unistd.h>

int readlink(const char *path,
                char *buf, size_t bufsiz);

#define _POSIX_C_SOURCE 200112L
#include <unistd.h>
```

```
ssize_t readlink(const char *__restrict__path,
                 char *__restrict__buf, size_t bufsiz);
```

General description

Places the contents of the symbolic link *path* in the buffer *buf*. The size of the buffer is set by *bufsiz*. The result stored in *buf* does not include a terminating NULL character.

If the buffer is too small to contain the value of the symbolic link, that value is truncated to the size of the buffer (*bufsiz*). If the value returned is the size of the buffer, use lstat() to determine the actual size of the symbolic link.

Returned value

If successful, when *bufsiz* is greater than 0, readlink() returns the number of bytes placed in the buffer. When *bufsiz* is 0 and readlink() completes successfully, it returns the number of bytes contained in the symbolic link and the buffer is not changed.

If the returned value is equal to *bufsiz*, you can determine the contents of the symbolic link with either lstat() or readlink(), with a 0 value for *bufsiz*.

If unsuccessful, readlink() returns -1 and sets errno to one of the following values:

Error Code
Description
EACCES

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

EINVAL

The named file is not a symbolic link.

EIO

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

ELOOP

A loop exists in symbolic links. This error is issued if more than POSIX_SYMLOOP symbolic links are encountered during resolution of the *path* argument.

ENAMETOOLONG

pathname is longer than **PATH_MAX** characters, or some component of *pathname* is longer than **NAME_MAX** characters while _POSIX_NO_TRUNC is in effect. For symbolic links, the length of the path name string substituted for a symbolic link exceeds **PATH_MAX**. The **PATH_MAX** and **NAME_MAX** values can be determined using pathconf().

ENOENT

The named file does not exist.

ENOTDIR

A component of the path prefix is not a directory.

Example

CELEBR05

```
/* CELEBR05 */
#define _POSIX_SOURCE 1
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#undef _POSIX_SOURCE
```

```
main() {
  char fn[]="readlink.file";
  char sl[]="readlink.symlink";
  char buf[30];
  int  fd;

  if ((fd = creat(fn, S_IWUSR)) < 0)
    perror("creat() error");
  else {
    close(fd);
    if (symlink(fn, sl) != 0)
      perror("symlink() error");
    else {
      if (readlink(sl, buf, sizeof(buf)) < 0)
        perror("readlink() error");
      else printf("readlink() returned '%s' for '%s'\n", buf, sl);

      unlink(sl);
    }
    unlink(fn);
  }
}
```

Output

```
readlink() returned 'readlink.file' for 'readlink.symlink'
```

Related information

- unistd.h – Implementation-specific functions
- lstat(), lstat64() – Get status of file or symbolic link
- stat(), stat64() – Get file information
- symlink() – Create a symbolic link to a path name

- unlink() – Remove a directory entry

Parent topic:

→ Library functions

Previous
readdir_r() – Read an entry from a directory

Next
readv() – Read data on a file or socket and store in a set of buffers