

bind(3p) — Linux manual page

[PROLOG](#) | [NAME](#) | [SYNOPSIS](#) | [DESCRIPTION](#) | [RETURN VALUE](#) | [ERRORS](#) | [EXAMPLES](#) |
[APPLICATION USAGE](#) | [RATIONALE](#) | [FUTURE DIRECTIONS](#) | [SEE ALSO](#) | [COPYRIGHT](#)

BIND(3P)**POSIX Programmer's Manual****BIND(3P)**

PROLOG

[top](#)

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

[top](#)

bind – bind a name to a socket

SYNOPSIS

[top](#)

```
#include <sys/socket.h>
```

```
int bind(int socket, const struct sockaddr *address,  
         socklen_t address_len);
```

DESCRIPTION

[top](#)

The `bind()` function shall assign a local socket address `address` to a socket identified by descriptor `socket` that has no local socket address assigned. Sockets created with the `socket()` function are initially unnamed; they are identified only by their address family.

The `bind()` function takes the following arguments:

- `socket` Specifies the file descriptor of the socket to be bound.
- `address` Points to a `sockaddr` structure containing the address to be bound to the socket. The length and format of the address depend on the address family of the socket.
- `address_len` Specifies the length of the `sockaddr` structure pointed to by the `address` argument.

The socket specified by `socket` may require the process to have appropriate privileges to use the `bind()` function.

If the address family of the socket is `AF_UNIX` and the pathname in `address` names a symbolic link, `bind()` shall fail and set `errno` to `[EADDRINUSE]`.

If the socket address cannot be assigned immediately and `O_NONBLOCK` is set for the file descriptor for the socket, `bind()` shall fail and set `errno` to `[EINPROGRESS]`, but the assignment request shall not be aborted, and the assignment shall be completed asynchronously. Subsequent calls to `bind()` for the same socket, before the assignment is completed, shall fail and set `errno` to `[EALREADY]`.

When the assignment has been performed asynchronously, `pselect()`,

`select()`, and `poll()` shall indicate that the file descriptor for the socket is ready for reading and writing.

RETURN VALUE [top](#)

Upon successful completion, `bind()` shall return 0; otherwise, -1 shall be returned and `errno` set to indicate the error.

ERRORS [top](#)

The `bind()` function shall fail if:

EADDRINUSE

The specified address is already in use.

EADDRNOTAVAIL

The specified address is not available from the local machine.

EAFNOSUPPORT

The specified address is not a valid address for the address family of the specified socket.

EALREADY

An assignment request is already in progress for the specified socket.

EBADF The `socket` argument is not a valid file descriptor.

EINPROGRESS

O_NONBLOCK is set for the file descriptor for the socket and the assignment cannot be immediately performed; the assignment shall be performed asynchronously.

EINVAL The socket is already bound to an address, and the protocol does not support binding to a new address; or the socket has been shut down.

ENOBUFS

Insufficient resources were available to complete the call.

ENOTSOCK

The *socket* argument does not refer to a socket.

EOPNOTSUPP

The socket type of the specified socket does not support binding to an address.

If the address family of the socket is AF_UNIX, then *bind()* shall fail if:

EACCES A component of the path prefix denies search permission, or the requested name requires writing in a directory with a mode that denies write permission.

EDESTADDRREQ or **EISDIR**

The *address* argument is a null pointer.

EIO An I/O error occurred.

ELOOP A loop exists in symbolic links encountered during resolution of the pathname in *address*.

ENAMETOOLONG

The length of a component of a pathname is longer than {NAME_MAX}.

ENOENT A component of the path prefix of the pathname in *address*

does not name an existing file or the pathname is an empty string.

ENOENT or **ENOTDIR**

The pathname in *address* contains at least one non-`<slash>` character and ends with one or more trailing `<slash>` characters. If the pathname without the trailing `<slash>` characters would name an existing file, an **[ENOENT]** error shall not occur.

ENOTDIR

A component of the path prefix of the pathname in *address* names an existing file that is neither a directory nor a symbolic link to a directory, or the pathname in *address* 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 directory nor a symbolic link to a directory.

EROFS The name would reside on a read-only file system.

The *bind()* function may fail if:

EACCES The specified address is protected and the current user does not have permission to bind to it.

EINVAL The *address_len* argument is not a valid length for the address family.

EISCONN

The socket is already connected.

ELOOP More than `{SYMLINK_MAX}` symbolic links were encountered during resolution of the pathname in *address*.

ENAMETOOLONG

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

The following sections are informative.

EXAMPLES [top](#)

The following code segment shows how to create a socket and bind it to a name in the AF_UNIX domain.

```
#define MY_SOCKET_PATH "/somepath"

int sfd;
struct sockaddr_un my_addr;

sfd = socket(AF_UNIX, SOCK_STREAM, 0);
if (sfd == -1)
    /* Handle error */;

memset(&my_addr, '\0', sizeof(struct sockaddr_un));
/* Clear structure */
my_addr.sun_family = AF_UNIX;
strncpy(my_addr.sun_path, MY_SOCKET_PATH, sizeof(my_addr.sun_path) - 1);

if (bind(sfd, (struct sockaddr *) &my_addr,
        sizeof(struct sockaddr_un)) == -1)
    /* Handle error */;
```

APPLICATION USAGE [top](#)

An application program can retrieve the assigned socket name with the *getsockname()* function.

RATIONALE [top](#)

None.

FUTURE DIRECTIONS [top](#)

None.

SEE ALSO [top](#)

[connect\(3p\)](#), [getsockname\(3p\)](#), [listen\(3p\)](#), [socket\(3p\)](#)

The Base Definitions volume of POSIX.1-2017, [sys_socket.h\(0p\)](#)

COPYRIGHT [top](#)

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1-2017, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of 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.kernel.org/doc/man-pages/reporting_bugs.html .

Pages that refer to this page: [netdb.h\(0p\)](#), [sys_socket.h\(0p\)](#), [sys_un.h\(0p\)](#), [accept\(3p\)](#), [connect\(3p\)](#), [getpeername\(3p\)](#), [getsockname\(3p\)](#), [getsockopt\(3p\)](#), [setsockopt\(3p\)](#), [socket\(3p\)](#)

HTML rendering created 2021-08-27 by [Michael Kerrisk](#), author of *The Linux Programming Interface*, maintainer of the Linux *man-pages* project.

For details of in-depth Linux/UNIX system programming training courses that I teach, look [here](#).

Hosting by [jambit GmbH](#).

