NAME

bind − bind a name to a socket

SYNOPSIS

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

int bind(int *sockfd***, struct sockaddr ****my_addr***, socklen_t** *addrlen***);**

DESCRIPTION

bind gives the socket *sockfd* the local address *my_addr*. *my_addr* is *addrlen* bytes long. Traditionally, this is called "assigning a name to a socket." When a socket is created with **socket**(2), it exists in a name space (address family) but has no name assigned.

It is normally necessary to assign a local address using **bind** before a **SOCK_STREAM** socket may receive connections (see **accept**(2)).

The rules used in name binding vary between address families. Consult the manual entries in Section 7 for detailed information. For **AF_INET** see **ip**(7), for **AF_UNIX** see **unix**(7), for **AF_APPLETALK** see **ddp**(7), for **AF_PACKET** see **packet**(7), for **AF_X25** see **x25**(7) and for **AF_NETLINK** see **netlink**(7).

RETURN VALUE

On success, zero is returned. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

EBADF
sockfd is not a valid descriptor.

EINVAL
The socket is already bound to an address. This may change in the future: see *linux/unix/sock.c* for details.

EACCES
The address is protected, and the user is not the super-user.

ENOTSOCK
Argument is a descriptor for a file, not a socket.

The following errors are specific to UNIX domain (**AF_UNIX**) sockets:

EINVAL
The *addrlen* is wrong, or the socket was not in the **AF_UNIX** family.

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

EFAULT
my_addr points outside the user's accessible address space.

ENAMETOOLONG
my_addr is too long.

ENOENT
The file does not exist.

ENOMEM
Insufficient kernel memory was available.

ENOTDIR
A component of the path prefix is not a directory.

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

ELOOP
Too many symbolic links were encountered in resolving *my_addr*.

BUGS

The transparent proxy options are not described.

CONFORMING TO

SVr4, 4.4BSD (the **bind** function first appeared in BSD 4.2). SVr4 documents additional **EADDRNOTAVAIL**, **EADDRINUSE**, and **ENOSR** general error conditions, and additional **EIO** and **EISDIR** Unix-domain error conditions.

NOTE

The third argument of **bind** is in reality an int (and this is what BSD 4.* and libc4 and libc5 have). Some POSIX confusion resulted in the present socklen_t. See also **accept**(2).

SEE ALSO

accept(2), **connect**(2), **listen**(2), **socket**(2), **getsockname**(2), **ip**(7), **socket**(7)