Rocky Enterprise Linux 9.2 Manual Pages on command 'bind.2'

$ man bind.2

BIND(2) Linux Programmer's Manual BIND(2)

NAME

bind - bind a name to a socket

SYNOPSIS

#include <sys/types.h> /* See NOTES */

#include <sys/socket.h>

int bind(int sockfd, const struct sockaddr *addr,

socklen_t addrlen);

DESCRIPTION

When a socket is created with socket(2), it exists in a name space (ad?

dress family) but has no address assigned to it. bind() assigns the

address specified by addr to the socket referred to by the file de?

scriptor sockfd. addrlen specifies the size, in bytes, of the address

structure pointed to by addr. Traditionally, this operation is called

?assigning a name to a socket?.

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_INET6, see ipv6(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).

The actual structure passed for the addr argument will depend on the address family. The sockaddr structure is defined as something like:

```
struct sockaddr {
    sa_family_t sa_family;
    char        sa_data[14];
}
```

The only purpose of this structure is to cast the structure pointer passed in addr in order to avoid compiler warnings. See EXAMPLES be? low.

RETURN VALUE

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

ERRORS

EACCES The address is protected, and the user is not the superuser.

EADDRINUSE

The given address is already in use.

EADDRINUSE

(Internet domain sockets) The port number was specified as zero in the socket address structure, but, upon attempting to bind to an ephemeral port, it was determined that all port numbers in the ephemeral port range are currently in use. See the discus? sion of /proc/sys/net/ipv4/ip_local_port_range ip(7).

EBADF sockfd is not a valid file descriptor.

EINVAL The socket is already bound to an address.

EINVAL addrlen is wrong, or addr is not a valid address for this socket's domain.

ENOTSOCK

The file descriptor sockfd does not refer to a socket.

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

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

(See also path_resolution(7).)

EADDRNOTAVAIL

A nonexistent interface was requested or the requested address

was not local.

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

ELOOP Too many symbolic links were encountered in resolving addr.

ENAMETOOLONG

addr is too long.

ENOENT A component in the directory prefix of the socket pathname does

not exist.

ENOMEM Insufficient kernel memory was available.

ENOTDIR

A component of the path prefix is not a directory.

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

CONFORMING TO

POSIX.1-2001, POSIX.1-2008, SVr4, 4.4BSD (bind() first appeared in

4.2BSD).

NOTES

POSIX.1 does not require the inclusion of <sys/types.h>, and this

header file is not required on Linux. However, some historical (BSD)

implementations required this header file, and portable applications

are probably wise to include it.

For background on the socklen_t type, see accept(2).

BUGS

The transparent proxy options are not described.

EXAMPLES

An example of the use of bind() with Internet domain sockets can be

found in getaddrinfo(3).

The following example shows how to bind a stream socket in the UNIX

(AF_UNIX) domain, and accept connections:

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

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#define MY_SOCK_PATH "/somepath"
#define LISTEN_BACKLOG 50
#define handle_error(msg) \
    do { perror(msg); exit(EXIT_FAILURE); } while (0)
int
main(int argc, char *argv[])
{
    int sfd, cfd;
    struct sockaddr_un my_addr, peer_addr;
    socklen_t peer_addr_size;
    sfd = socket(AF_UNIX, SOCK_STREAM, 0);
    if (sfd == -1)
        handle_error("socket");
    memset(&my_addr, 0, sizeof(my_addr));
                        /* Clear structure */
    my_addr.sun_family = AF_UNIX;
    strncpy(my_addr.sun_path, MY_SOCK_PATH,
            sizeof(my_addr.sun_path) - 1);
    if (bind(sfd, (struct sockaddr *) &my_addr,
            sizeof(my_addr)) == -1)
        handle_error("bind");
    if (listen(sfd, LISTEN_BACKLOG) == -1)
        handle_error("listen");
    /* Now we can accept incoming connections one
       at a time using accept(2) */
    peer_addr_size = sizeof(peer_addr);
    cfd = accept(sfd, (struct sockaddr *) &peer_addr,
                 &peer_addr_size);
    if (cfd == -1)
        handle_error("accept");
```

```
        /* Code to deal with incoming connection(s)... */

        /* When no longer required, the socket pathname, MY_SOCK_PATH
           should be deleted using unlink(2) or remove(3) */
    }
```

SEE ALSO

accept(2), connect(2), getsockname(2), listen(2), socket(2), getad?
drinfo(3), getifaddrs(3), ip(7), ipv6(7), path_resolution(7),
socket(7), unix(7)

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.