

NAME

connect – initiate a connection on a socket

SYNOPSIS

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

```
int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);
```

DESCRIPTION

The file descriptor *sockfd* must refer to a socket. If the socket is of type **SOCK_DGRAM** then the *serv_addr* address is the address to which datagrams are sent by default, and the only address from which datagrams are received. If the socket is of type **SOCK_STREAM** or **SOCK_SEQPACKET**, this call attempts to make a connection to another socket. The other socket is specified by *serv_addr*, which is an address (of length *addrlen*) in the communications space of the socket. Each communications space interprets the *serv_addr* parameter in its own way.

Generally, connection-based protocol sockets may successfully **connect** only once; connectionless protocol sockets may use **connect** multiple times to change their association. Connectionless sockets may dissolve the association by connecting to an address with the *sa_family* member of **sockaddr** set to **AF_UNSPEC**.

RETURN VALUE

If the connection or binding succeeds, zero is returned. On error, **-1** is returned, and *errno* is set appropriately.

ERRORS

The following are general socket errors only. There may be other domain-specific error codes.

EBADF

The file descriptor is not a valid index in the descriptor table.

EFAULT

The socket structure address is outside the user's address space.

ENOTSOCK

The file descriptor is not associated with a socket.

EISCONN

The socket is already connected.

ECONNREFUSED

No one listening on the remote address.

ETIMEDOUT

Timeout while attempting connection. The server may be too busy to accept new connections. Note that for IP sockets the timeout may be very long when syncookies are enabled on the server.

ENETUNREACH

Network is unreachable.

EADDRINUSE

Local address is already in use.

EINPROGRESS

The socket is non-blocking and the connection cannot be completed immediately. It is possible to **select(2)** or **poll(2)** for completion by selecting the socket for writing. After **select** indicates writability, use **getsockopt(2)** to read the **SO_ERROR** option at level **SOL_SOCKET** to determine whether **connect** completed successfully (**SO_ERROR** is zero) or unsuccessfully (**SO_ERROR** is one of the usual error codes listed here, explaining the reason for the failure).

EALREADY

The socket is non-blocking and a previous connection attempt has not yet been completed.

EAGAIN

No more free local ports or insufficient entries in the routing cache. For **PF_INET** see the **net.ipv4.ip_local_port_range** sysctl in **ip(7)** on how to increase the number of local ports.

EAFNOSUPPORT

The passed address didn't have the correct address family in its *sa_family* field.

EACCES, EPERM

The user tried to connect to a broadcast address without having the socket broadcast flag enabled or the connection request failed because of a local firewall rule.

CONFORMING TO

SVr4, 4.4BSD (the **connect** function first appeared in BSD 4.2). SVr4 documents the additional general error codes **EADDRNOTAVAIL**, **EINVAL**, **EAFNOSUPPORT**, **EALREADY**, **EINTR**, **EPROTOTYPE**, and **ENOSR**. It also documents many additional error conditions not described here.

NOTE

The third argument of **connect** 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. The draft standard has not been adopted yet, but glibc2 already follows it and also has socklen_t. See also **accept(2)**.

BUGS

Unconnecting a socket by calling **connect** with a **AF_UNSPEC** address is not yet implemented.

SEE ALSO

accept(2), **bind(2)**, **listen(2)**, **socket(2)**, **getsockname(2)**