NAME

connect - initiate a connection on a socket

SYNOPSIS

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

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

DESCRIPTION

The **connect**() system call connects the socket referred to by the file descriptor *sockfd* to the address specified by *addr*. The*addrlen* ar gument specifies the size of *addr*. The format of the address in*addr* is deter - mined by the address space of the socket *sockfd*; see socket(2) for further details.

If the socket *sockfd* is of type **SOCK_DGRAM**, then *addr* 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 the socket that is bound to the address specified by *addr*.

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** (supported on Linux since kernel 2.2).

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.

EACCES
For UNIX domain sockets, which are identified by pathname: Write permission is denied on the socket file, or search permission is denied for one of the directories in the path prefix. (See also path_resolution(7).)

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.

EADDRINUSE
Local address is already in use.

EADDRNOTAVAIL
(Internet domain sockets) The socket referred to by *sockfd* had not previously been bound to an address and, upon attempting to bind it to an ephemeral port, it was determined that all port numbers in the ephemeral port range are currently in use. See the discussion of */proc/sys/net/ipv4/ip_local_port_range* in ip(7).

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

EAGAIN
Insufficient entries in the routing cache.

EALREADY
The socket is nonblocking and a previous connection attempt has not yet been completed.

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

ECONNREFUSED
No-one listening on the remote address.

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

EINPROGRESS
The socket is nonblocking 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(2) 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).

EINTR
The system call was interrupted by a signal that was caught; see signal(7).

EISCONN
The socket is already connected.

ENETUNREACH
Network is unreachable.

ENOTSOCK
The file descriptor is not associated with a socket.

EPROTOTYPE
The socket type does not support the requested communications protocol. This error can occur, for example, on an attempt to connect a UNIX domain datagram socket to a stream socket.

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.

CONFORMING TO

SVr4, 4.4BSD, (the **connect**() function first appeared in 4.2BSD), POSIX.1-2001.

NOTES

POSIX.1-2001 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.

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

If **connect()** fails, consider the state of the socket as unspecified. Portable applications should close the socket and create a new one for reconnecting.

EXAMPLE

An example of the use of **connect**() is shown in getaddrinfo(3).

SEE ALSO

accept(2), bind(2), getsockname(2), listen(2), socket(2), path_resolution(7)

COLOPHON

This page is part of release 3.74 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 http://www.kernel.org/doc/man-pages/.