NAME

accept, **accept4** **-** accept a connection on a socket

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

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

int
accept(*int s*, *struct sockaddr * restrict addr*, *socklen_t * restrict addrlen*);

int
accept4(*int s*, *struct sockaddr * restrict addr*, *socklen_t * restrict addrlen*, *int flags*);

DESCRIPTION

The argument *s* is a socket that has been created with socket(2), bound to an address with bind(2), and is listening for connections after a listen(2). The **accept**() system call extracts the first connection request on the queue of pending connections, creates a new socket, and allocates a new file descriptor for the socket which inherits the state of the O_NONBLOCK and O_ASYNC properties and the destination of SIGIO and SIGURG signals from the original socket *s*.

The **accept4**() system call is similar, but the O_NONBLOCK property of the new socket is instead determined by the SOCK_NONBLOCK flag in the *flags* argument, the O_ASYNC property is cleared, the signal destination is cleared and the close-on-exec flag on the new file descriptor can be set via the SOCK_CLOEXEC flag in the *flags* argument.

If no pending connections are present on the queue, and the original socket is not marked as non-blocking, **accept**() blocks the caller until a connection is present. If the original socket is marked non-blocking and no pending connections are present on the queue, **accept**() returns an error as described below. The accepted socket may not be used to accept more connections. The original socket *s* remains open.

The argument *addr* is a result argument that is filled-in with the address of the connecting entity, as known to the communications layer. The exact format of the *addr* argument is determined by the domain in which the communication is occurring. A null pointer may be specified for *addr* if the address information is not desired; in this case, *addrlen* is not used and should also be null. Otherwise, the *addrlen* argument is a value-result argument; it should initially contain the amount of space pointed to by *addr*; on return it will contain the actual length (in bytes) of the address returned. This call is used with connection-based socket types, currently with SOCK_STREAM.

It is possible to select(2) a socket for the purposes of doing an **accept**() by selecting it for read.

For certain protocols which require an explicit confirmation, such as ISO or DATAKIT, **accept**() can be thought of as merely dequeueing the next connection request and not implying confirmation. Confirmation can be implied by a normal read or write on the new file descriptor, and rejection can be implied by closing the new socket.

For some applications, performance may be enhanced by using an accept_filter(9) to pre-process incoming connections.

When using **accept**(), portable programs should not rely on the O_NONBLOCK and O_ASYNC properties and the signal destination being inherited, but should set them explicitly using fcntl(2); **accept4**() sets these properties consistently, but may not be fully portable across UNIX platforms.

RETURN VALUES

These calls return -1 on error. If they succeed, they return a non-negative integer that is a descriptor for the accepted socket.

ERRORS

The **accept**() and **accept4**() system calls will fail if:

[EBADF]	The descriptor is invalid.
[EINTR]	The **accept**() operation was interrupted.
[EMFILE]	The per-process descriptor table is full.
[ENFILE]	The system file table is full.
[ENOTSOCK]	The descriptor references a file, not a socket.
[EINVAL]	listen(2) has not been called on the socket descriptor.
[EFAULT]	The *addr* argument is not in a writable part of the user address space.
[EWOULDBLOCK] or [EAGAIN]	The socket is marked non-blocking and no connections are present to be accepted.

[ECONNABORTED]
A connection arrived, but it was closed while waiting on the listen queue.

The **accept4**() system call will also fail if:

[EINVAL] The *flags* argument is invalid.

SEE ALSO

bind(2), connect(2), getpeername(2), getsockname(2), listen(2), select(2), socket(2), accept_filter(9)

HISTORY

The **accept**() system call appeared in 4.2BSD.

The **accept4**() system call appeared in FreeBSD 10.0.