

accept(3p) — Linux manual page

[PROLOG](#) | [NAME](#) | [SYNOPSIS](#) | [DESCRIPTION](#) | [RETURN VALUE](#) | [ERRORS](#) | [EXAMPLES](#) |
[APPLICATION USAGE](#) | [RATIONALE](#) | [FUTURE DIRECTIONS](#) | [SEE ALSO](#) | [COPYRIGHT](#)

ACCEPT(3P)**POSIX Programmer's Manual****ACCEPT(3P)****PROLOG**[top](#)

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME[top](#)

accept – accept a new connection on a socket

SYNOPSIS[top](#)

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

```
int accept(int socket, struct sockaddr *restrict address,  
           socklen_t *restrict address_len);
```

DESCRIPTION[top](#)

The `accept()` function shall extract the first connection on the queue of pending connections, create a new socket with the same socket type protocol and address family as the specified socket, and allocate a new file descriptor for that socket. The file descriptor shall be allocated as described in *Section 2.14, Descriptor Allocation*.

The `accept()` function takes the following arguments:

- socket* Specifies a socket that was created with `socket()`, has been bound to an address with `bind()`, and has issued a successful call to `listen()`.
- address* Either a null pointer, or a pointer to a **sockaddr** structure where the address of the connecting socket shall be returned.
- address_len* Either a null pointer, if *address* is a null pointer, or a pointer to a object which on input specifies the length of the supplied **sockaddr** structure, and on output specifies the length of the stored address.

If *address* is not a null pointer, the address of the peer for the accepted connection shall be stored in the **sockaddr** structure pointed to by *address*, and the length of this address shall be stored in the object pointed to by *address_len*.

If the actual length of the address is greater than the length of the supplied structure, the stored address shall be truncated.

If the protocol permits connections by unbound clients, and the peer is not bound, then the value stored in the object pointed to by *address* is unspecified.

If the listen queue is empty of connection requests and `O_NONBLOCK` is not set on the file descriptor for the socket, `accept()` shall block until a connection is present. If the `listen()` queue is empty of connection requests and `O_NONBLOCK` is set on the file descriptor for the socket, `accept()` shall fail and set `errno` to **[EAGAIN]** or **[EWOULDBLOCK]**.

The accepted socket cannot itself accept more connections. The original socket remains open and can accept more connections.

RETURN VALUE [top](#)

Upon successful completion, `accept()` shall return the non-negative file descriptor of the accepted socket. Otherwise, `-1` shall be returned, `errno` shall be set to indicate the error, and any object pointed to by `address_len` shall remain unchanged.

ERRORS [top](#)

The `accept()` function shall fail if:

EAGAIN or **EWOULDBLOCK**

`O_NONBLOCK` is set for the socket file descriptor and no connections are present to be accepted.

EBADF The `socket` argument is not a valid file descriptor.

ECONNABORTED

A connection has been aborted.

EINTR `accept()` function was interrupted by a signal that was caught before a valid connection arrived.

EINVAL *socket* is not accepting connections.

EMFILE All file descriptors available to the process are currently open.

ENFILE The maximum number of file descriptors in the system are already open.

ENOBUFFS

No buffer space is available.

ENOMEM There was insufficient memory available to complete the operation.

ENOTSOCK

The *socket* argument does not refer to a socket.

EOPNOTSUPP

The socket type of the specified socket does not support accepting connections.

The *accept()* function may fail if:

EPROTO A protocol error has occurred; for example, the STREAMS protocol stack has not been initialized.

The following sections are informative.

EXAMPLES [top](#)

None.

APPLICATION USAGE [top](#)

When a connection is available, *select()* indicates that the file descriptor for the socket is ready for reading.

RATIONALE [top](#)

None.

FUTURE DIRECTIONS [top](#)

None.

SEE ALSO [top](#)

Section 2.14, File Descriptor Allocation, [bind\(3p\)](#), [connect\(3p\)](#), [listen\(3p\)](#), [socket\(3p\)](#)

The Base Definitions volume of POSIX.1-2017, [sys_socket.h\(0p\)](#)

COPYRIGHT [top](#)

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1-2017, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at <http://www.opengroup.org/unix/online.html> .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of

the source files to man page format. To report such errors, see
https://www.kernel.org/doc/man-pages/reporting_bugs.html .

IEEE/The Open Group

2017

ACCEPT(3P)

Pages that refer to this page: [sys_socket.h\(0p\)](#), [connect\(3p\)](#), [getpeername\(3p\)](#),
[getsockname\(3p\)](#), [listen\(3p\)](#), [pselect\(3p\)](#), [socket\(3p\)](#)

HTML rendering created 2021-08-27 by [Michael Kerrisk](#), author of *The Linux Programming Interface*, maintainer of the *Linux man-pages*.

For details of in-depth **Linux/UNIX system programming training courses** that I teach, look [here](#).

Hosting by [jambit GmbH](#).

