

close() — Close a File

Standards

Standards / Extensions	C or C++	Dependencies
POSIX.1 XPG4 XPG4.2 Single UNIX Specification, Version 3	both	

Format

```
#define _POSIX_SOURCE
#include <unistd.h>

int close(int filde);
```

X/Open

```
#define _XOPEN_SOURCE_EXTENDED 1
#include <unistd.h>

int close(int socket);
```

Berkeley Sockets

```
#define _OE_SOCKETS
#include <unistd.h>

int close(int socket);
```

General Description

Closes a file descriptor, *filde*. This frees the file descriptor to be returned by future `open()` calls and other calls that create file descriptors. The *filde* argument must represent a hierarchical file system (HFS) file.

When the last open file descriptor for a file is closed, the file itself is closed. If the file's link count is 0 at that time, its space is freed and the file becomes inaccessible.

When the last open file descriptor for a pipe or FIFO file is closed, any data remaining in the pipe or FIFO file is discarded. See [Language Environment element](#) for more information about using POSIX support.

`close()` unlocks (removes) all outstanding record locks that a process has on the associated file.

Behavior for Sockets

`close()` call shuts down the socket associated with the socket descriptor *socket*, and frees resources allocated to the socket. If *socket* refers to an open TCP connection, the connection is closed. If a stream socket is closed when there is input data queued, the TCP connection is reset rather than being cleanly closed.

Parameter

Description

socket

The descriptor of the socket to be closed.

Note:

All sockets should be closed before the end of your process. You should issue a `shutdown()` call before you issue a `close()` call for a socket.

For `AF_INET` and `AF_INET6` stream sockets (`SOCK_STREAM`) using `SO_LINGER` socket option, the socket does not immediately end if data is still present when a `close` is issued. The following structure is used to set or unset this option, and it can be found in **sys/socket.h**.

```
struct linger {
    int l_onoff;      /* zero=off, nonzero=on */
    int l_linger;     /* time is seconds to linger */
};
```

If the `l_onoff` switch is nonzero, the system attempts to deliver any unsent messages. If a linger time is specified, the system waits for *n* seconds before flushing the data and terminating the socket.

For `AF_UNIX`, when closing sockets that were bound, you should also use `unlink()` to delete the file created at `bind()` time.

Special Behavior for XPG4.2

If a STREAMS-based *fildev* is closed and the calling process was previously registered to receive a `SIGPOL` signal for events associated with that STREAM, the calling process will be unregistered for events associated with the STREAM. The last `close()` for a STREAM causes the STREAM associated with *fildev* to be dismantled. If `O_NONBLOCK` is not set and there have been no signals posted for the STREAM, and if there is data on the module's write queue, `close()` waits for an unspecified time (for each module and driver) for any output to drain before dismantling the STREAM. The time delay can be changed using an `I_SETCLTIME` `ioctl()` request. If the `O_NONBLOCK` flag is set, or if there are any pending signals, `close()` does not wait for output to drain, and dismantles the STREAM immediately.

Note:

z/OS UNIX services do not supply any STREAMS devices or pseudodevices. See [open\(\) — Open a File](#) for more information.

If *fildev* refers to the master side of a pseudoterminal, a SIGHUP signal is sent to the process group, if any, for which the slave side of the pseudoterminal is the controlling terminal.

If *fildev* refers to the slave side of a pseudoterminal, a zero-length message will be sent to the master.

If *fildev* refers to a socket, close() causes the socket to be destroyed. If the socket is connection-oriented and the SO_LINGER option is set for the socket and the socket has untransmitted data, then close() will block for up to the current linger interval until all data is transmitted.

Returned Value

If successful, close() returns 0.

If unsuccessful, close() returns -1 and sets errno to one of the following values:

Error Code

Description

EAGAIN

The call did not complete because the specified socket descriptor is currently being used by another thread in the same process.

For example, in a multithreaded environment, close() fails and returns EAGAIN when the following sequence of events occurs (1) thread is blocked in a read() or select() call on a given file or socket descriptor and (2) another thread issues a simultaneous close() call for the same descriptor.

EBADF

fildev is not a valid open file descriptor, or the *socket* parameter is not a valid socket descriptor.

EBUSY

The file cannot be closed because it is blocked.

EINTR

close() was interrupted by a signal. The file may or may not be closed.

EIO

Added for XPG4.2: An I/O error occurred while reading from or writing to the file system.

ENXIO

fildev does not exist. The minor number for the file is incorrect.