NAME

close − close a file descriptor

SYNOPSIS

#include <unistd.h>

int close(int *fd***);**

DESCRIPTION

close() closes a file descriptor, so that it no longer refers to any file and may be reused. Any record locks (see **fcntl**(2)) held on the file it was associated with, and owned by the process, are removed (regardless of the file descriptor that was used to obtain the lock).

If *fd* is the last file descriptor referring to the underlying open file description (see **open**(2)), the resources associated with the open file description are freed; if the descriptor was the last reference to a file which has been removed using **unlink**(2) the file is deleted.

RETURN VALUE

close() returns zero on success. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

EBADF
fd isn't a valid open file descriptor.

EINTR
The **close**() call was interrupted by a signal; see **signal**(7).

EIO An I/O error occurred.

CONFORMING TO

SVr4, 4.3BSD, POSIX.1-2001.

NOTES

Not checking the return value of **close**() is a common but nevertheless serious programming error. It is quite possible that errors on a previous **write**(2) operation are first reported at the final **close**(). Not checking the return value when closing the file may lead to silent loss of data. This can especially be observed with NFS and with disk quota.

A successful close does not guarantee that the data has been successfully saved to disk, as the kernel defers writes. It is not common for a file system to flush the buffers when the stream is closed. If you need to be sure that the data is physically stored use **fsync**(2). (It will depend on the disk hardware at this point.)

It is probably unwise to close file descriptors while they may be in use by system calls in other threads in the same process. Since a file descriptor may be re-used, there are some obscure race conditions that may cause unintended side effects.

When dealing with sockets, you have to be sure that there is no **recv**(2) still blocking on it on another thread, otherwise it might block forever, since no more messages will be sent via the socket. Be sure to use **shutdown**(2) to shut down all parts the connection before closing the socket.

SEE ALSO

fcntl(2), **fsync**(2), **open**(2), **shutdown**(2), **unlink**(2), **fclose**(3)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.