## NAME

**select** - synchronous I/O multiplexing

## LIBRARY

Standard C Library (libc, -lc)

## SYNOPSIS

**#include <sys/select.h>**

*int*
**select**(*int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout*);

**FD_SET**(*fd, &fdset*);

**FD_CLR**(*fd, &fdset*);

**FD_ISSET**(*fd, &fdset*);

**FD_ZERO**(*&fdset*);

## DESCRIPTION

The **select**() system call examines the I/O descriptor sets whose addresses are passed in *readfds*, *writefds*, and *exceptfds* to see if some of their descriptors are ready for reading, are ready for writing, or have an exceptional condition pending, respectively. The only exceptional condition detectable is out-of-band data received on a socket. The first *nfds* descriptors are checked in each set; i.e., the descriptors from 0 through *nfds*-1 in the descriptor sets are examined. On return, **select**() replaces the given descriptor sets with subsets consisting of those descriptors that are ready for the requested operation. The **select**() system call returns the total number of ready descriptors in all the sets.

The descriptor sets are stored as bit fields in arrays of integers. The following macros are provided for manipulating such descriptor sets: **FD_ZERO**(*&fdset*) initializes a descriptor set *fdset* to the null set. **FD_SET**(*fd, &fdset*) includes a particular descriptor *fd* in *fdset*. **FD_CLR**(*fd, &fdset*) removes *fd* from *fdset*. **FD_ISSET**(*fd, &fdset*) is non-zero if *fd* is a member of *fdset*, zero otherwise. The behavior of these macros is undefined if a descriptor value is less than zero or greater than or equal to FD_SETSIZE, which is normally at least equal to the maximum number of descriptors supported by the system.

If *timeout* is not a null pointer, it specifies the maximum interval to wait for the selection to complete. System activity can lengthen the interval by an indeterminate amount.

If *timeout* is a null pointer, the select blocks indefinitely.

To effect a poll, the *timeout* argument should not be a null pointer, but it should point to a zero-valued timeval structure.

Any of *readfds*, *writefds*, and *exceptfds* may be given as null pointers if no descriptors are of interest.

## RETURN VALUES

The **select**() system call returns the number of ready descriptors that are contained in the descriptor sets, or -1 if an error occurred. If the time limit expires, **select**() returns 0. If **select**() returns with an error, including one due to an interrupted system call, the descriptor sets will be unmodified.

## ERRORS

An error return from **select**() indicates:

[EBADF] One of the descriptor sets specified an invalid descriptor.

[EFAULT] One of the arguments *readfds*, *writefds*, *exceptfds*, or *timeout* points to an invalid address.

[EINTR] A signal was delivered before the time limit expired and before any of the selected events occurred.

[EINVAL] The specified time limit is invalid. One of its components is negative or too large.

[EINVAL] The *nfds* argument was invalid.

## SEE ALSO

accept(2), connect(2), getdtablesize(2), gettimeofday(2), kqueue(2), poll(2), pselect(2), read(2), recv(2), send(2), write(2), clocks(7)

## NOTES

The default size of FD_SETSIZE is currently 1024. In order to accommodate programs which might potentially use a larger number of open files with **select**(), it is possible to increase this size by having the program define FD_SETSIZE before the inclusion of any header which includes <*sys/types.h*>.

If *nfds* is greater than the number of open files, **select**() is not guaranteed to examine the unused file descriptors. For historical reasons, **select**() will always examine the first 256 descriptors.

## STANDARDS

The **select**() system call and **FD_CLR**(), **FD_ISSET**(), **FD_SET**(), and **FD_ZERO**() macros conform with IEEE Std 1003.1-2001 ("POSIX.1").

## HISTORY

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

## BUGS

Version 2 of the Single UNIX Specification ("SUSv2") allows systems to modify the original timeout in place. Thus, it is unwise to assume that the timeout value will be unmodified by the **select**() system call. FreeBSD does not modify the return value, which can cause problems for applications ported from other systems.