NAME

poll - synchronous I/O multiplexing

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <poll.h>

int
poll(*struct pollfd fds[], nfds_t nfds, int timeout*);

int
ppoll(*struct pollfd fds[], nfds_t nfds, const struct timespec * restrict timeout,*
*const sigset_t * restrict newsigmask*);

DESCRIPTION

The **poll**() system call examines a set of file descriptors to see if some of them are ready for I/O. The *fds* argument is a pointer to an array of pollfd structures as defined in <*poll.h*> (shown below). The *nfds* argument determines the size of the *fds* array.

```
struct pollfd {
   int    fd;       /* file descriptor */
   short  events;   /* events to look for */
   short  revents;  /* events returned */
};
```

The fields of *struct pollfd* are as follows:

fd — File descriptor to poll. If fd is equal to -1 then *revents* is cleared (set to zero), and that pollfd is not checked.

events — Events to poll for. (See below.)

revents — Events which may occur. (See below.)

The event bitmasks in *events* and *revents* have the following bits:

POLLIN — Data other than high priority data may be read without blocking.

POLLRDNORM	Normal data may be read without blocking.
POLLRDBAND	Data with a non-zero priority may be read without blocking.
POLLPRI	High priority data may be read without blocking.
POLLOUT	
POLLWRNORM	Normal data may be written without blocking.
POLLWRBAND	Data with a non-zero priority may be written without blocking.
POLLERR	An exceptional condition has occurred on the device or socket. This flag is always checked, even if not present in the *events* bitmask.
POLLHUP	The device or socket has been disconnected. This flag is always checked, even if not present in the *events* bitmask. Note that POLLHUP and POLLOUT should never be present in the *revents* bitmask at the same time.
POLLRDHUP	Remote peer closed connection, or shut down writing. Unlike POLLHUP, POLLRDHUP must be present in the *events* bitmask to be reported. Applies only to stream sockets.
POLLNVAL	The file descriptor is not open, or in capability mode the file descriptor has insufficient rights. This flag is always checked, even if not present in the *events* bitmask.

If *timeout* is neither zero nor INFTIM (-1), it specifies a maximum interval to wait for any file descriptor to become ready, in milliseconds. If *timeout* is INFTIM (-1), the poll blocks indefinitely. If *timeout* is zero, then **poll**() will return without blocking.

The **ppoll**() system call, unlike **poll**(), is used to safely wait until either a set of file descriptors becomes ready or until a signal is caught. The *fds* and *nfds* arguments are identical to the analogous arguments of **poll**(). The *timeout* argument in **ppoll**() points to a *const struct timespec* which is defined in <*sys/timespec.h*> (shown below) rather than the *int timeout* used by **poll**(). A null pointer may be passed to indicate that **ppoll**() should wait indefinitely. Finally, *newsigmask* specifies a signal mask which is set while waiting for input. When **ppoll**() returns, the original signal mask is restored.

```
struct timespec {
        time_t  tv_sec;         /* seconds */
```

```
        long    tv_nsec;        /* and nanoseconds */
};
```

RETURN VALUES

The **poll**() system call returns the number of descriptors that are ready for I/O, or -1 if an error occurred. If the time limit expires, **poll**() returns 0. If **poll**() returns with an error, including one due to an interrupted system call, the *fds* array will be unmodified.

COMPATIBILITY

This implementation differs from the historical one in that a given file descriptor may not cause **poll**() to return with an error. In cases where this would have happened in the historical implementation (e.g. trying to poll a revoke(2)ed descriptor), this implementation instead copies the *events* bitmask to the *revents* bitmask. Attempting to perform I/O on this descriptor will then return an error. This behaviour is believed to be more useful.

ERRORS

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

[EFAULT]	The *fds* argument points outside the process's allocated address space.
[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 number of pollfd structures specified by *nfds* exceeds the system tunable *kern.maxfilesperproc* and FD_SETSIZE.

SEE ALSO

accept(2), connect(2), kqueue(2), pselect(2), read(2), recv(2), select(2), send(2), write(2)

STANDARDS

The **poll**() function conforms to IEEE Std 1003.1-2001 ("POSIX.1"). The **ppoll**() is not specified by POSIX. The POLLRDHUP flag is not specified by POSIX, but is compatible with Linux and illumos.

HISTORY

The **poll**() function appeared in AT&T System V UNIX. This manual page and the core of the implementation was taken from NetBSD. The **ppoll**() function first appeared in FreeBSD 11.0

BUGS

The distinction between some of the fields in the *events* and *revents* bitmasks is really not useful without STREAMS. The fields are defined for compatibility with existing software.