NAME

poll, ppoll − wait for some event on a file descriptor

SYNOPSIS

```
#include <poll.h>

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

#define _GNU_SOURCE
#include <poll.h>

int ppoll(struct pollfd * fds, nfds_t nfds,
    const struct timespec *timeout, const sigset_t *sigmask);
```

DESCRIPTION

poll() performs a similar task to **select**(2): it waits for one of a set of file descriptors to become ready to perform I/O.

The set of file descriptors to be monitored is specified in the *fds* argument, which is an array of *nfds* structures of the following form:

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

The field *fd* contains a file descriptor for an open file.

The field *events* is an input parameter, a bit mask specifying the events the application is interested in.

The field *revents* is an output parameter, filled by the kernel with the events that actually occurred. The bits returned in *revents* can include any of those specified in *events*, or one of the values **POLLERR**, **POLLHUP**, or **POLLNVAL**. (These three bits are meaningless in the *events* field, and will be set in the *revents* field whenever the corresponding condition is true.)

If none of the events requested (and no error) has occurred for any of the file descriptors, then **poll**() blocks until one of the events occurs.

The *timeout* argument specifies an upper limit on the time for which **poll**() will block, in milliseconds. Specifying a negative value in *timeout* means an infinite timeout.

The bits that may be set/returned in *events* and *revents* are defined in <*poll.h*>:

POLLIN
There is data to read.

POLLPRI
There is urgent data to read (e.g., out-of-band data on TCP socket; pseudo-terminal master in packet mode has seen state change in slave).

POLLOUT
Writing now will not block.

POLLRDHUP (since Linux 2.6.17)
Stream socket peer closed connection, or shut down writing half of connection. The **_GNU_SOURCE** feature test macro must be defined in order to obtain this definition.

POLLERR
Error condition (output only).

POLLHUP
Hang up (output only).

POLLNVAL
Invalid request: *fd* not open (output only).

When compiling with **_XOPEN_SOURCE** defined, one also has the following, which convey no further information beyond the bits listed above:

POLLRDNORM
Equivalent to **POLLIN**.

POLLRDBAND
Priority band data can be read (generally unused on Linux).

POLLWRNORM
Equivalent to **POLLOUT**.

POLLWRBAND
Priority data may be written.

Linux also knows about, but does not use **POLLMSG**.

ppoll()
The relationship between **poll**() and **ppoll**() is analogous to the relationship between **select**(2) and **pselect**(2): like **pselect**(2), **ppoll**() allows an application to safely wait until either a file descriptor becomes ready or until a signal is caught.

Other than the difference in the *timeout* argument, the following **ppoll**() call:

```
ready = ppoll(&fds, nfds, timeout, &sigmask);
```

is equivalent to *atomically* executing the following calls:

```
sigset_t origmask;

sigprocmask(SIG_SETMASK, &sigmask, &origmask);
ready = poll(&fds, nfds, timeout);
sigprocmask(SIG_SETMASK, &origmask, NULL);
```

See the description of **pselect**(2) for an explanation of why **ppoll**() is necessary.

If the *sigmask* argument is specified as NULL, then no signal mask manipulation is performed (and thus **ppoll**() differs from **poll**() only in the precision of the *timeout* argument).

The *timeout* argument specifies an upper limit on the amount of time that **ppoll**() will block. This argument is a pointer to a structure of the following form:

```
struct timespec {
   long    tv_sec;        /* seconds */
   long    tv_nsec;       /* nanoseconds */
};
```

If *timeout* is specified as NULL, then **ppoll**() can block indefinitely.

RETURN VALUE

On success, a positive number is returned; this is the number of structures which have non-zero *revents* fields (in other words, those descriptors with events or errors reported). A value of 0 indicates that the call timed out and no file descriptors were ready. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

EFAULT
The array given as argument was not contained in the calling program's address space.

EINTR
A signal occurred before any requested event; see **signal**(7).

EINVAL
The *nfds* value exceeds the **RLIMIT_NOFILE** value.

ENOMEM
There was no space to allocate file descriptor tables.

VERSIONS

The **poll**() system call was introduced in Linux 2.1.23. The **poll**() library call was introduced in libc 5.4.28 (and provides emulation using select(2) if your kernel does not have a **poll**() system call).

The **ppoll**() system call was added to Linux in kernel 2.6.16. The **ppoll**() library call was added in glibc 2.4.

CONFORMING TO

poll() conforms to POSIX.1-2001. **ppoll**() is Linux-specific.

NOTES

Some implementations define the non-standard constant **INFTIM** with the value −1 for use as a *timeout*. This constant is not provided in glibc.

Linux Notes

The Linux **ppoll**() system call modifies its *timeout* argument. However, the glibc wrapper function hides this behavior by using a local variable for the timeout argument that is passed to the system call. Thus, the glibc **ppoll**() function does not modify its *timeout* argument.

BUGS

See the discussion of spurious readiness notifications under the BUGS section of **select**(2).

SEE ALSO

select(2), **select_tut**(2), **feature_test_macros**(7), **time**(7)

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/.