NAME

epoll_wait, epoll_pwait − wait for an I/O event on an epoll file descriptor

SYNOPSIS

#include <sys/epoll.h>

int epoll_wait(int *epfd*, **struct epoll_event ***events,
int *maxevents*, **int** *timeout*);
int epoll_pwait(int *epfd*, **struct epoll_event ***events,
int *maxevents*, **int** *timeout*,
const sigset_t ***sigmask);**

DESCRIPTION

The **epoll_wait**() system call waits for events on the **epoll** instance referred to by the file descriptor *epfd*. The memory area pointed to by *events* will contain the events that will be available for the caller. Up to *maxevents* are returned by **epoll_wait**(). The *maxevents* argument must be greater than zero.

The call waits for a maximum time of *timeout* milliseconds. Specifying a *timeout* of −1 makes **epoll_wait**() wait indefinitely, while specifying a *timeout* equal to zero makes **epoll_wait**() to return immediately even if no events are available (return code equal to zero).

The *struct epoll_event* is defined as :

```
typedef union epoll_data {
   void    *ptr;
   int      fd;
   uint32_t u32;
   uint64_t u64;
} epoll_data_t;

struct epoll_event {
   uint32_t     events;    /* Epoll events */
   epoll_data_t data;      /* User data variable */
};
```

The *data* of each returned structure will contain the same data the user set with an **epoll_ctl**(2) (**EPOLL_CTL_ADD**,**EPOLL_CTL_MOD**) while the *events* member will contain the returned event bit field.

epoll_pwait()

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

The following **epoll_pwait**() call:

```
ready = epoll_pwait(epfd, &events, maxevents, timeout, &sigmask);
```

is equivalent to *atomically* executing the following calls:

```
sigset_t origmask;

sigprocmask(SIG_SETMASK, &sigmask, &origmask);
ready = epoll_wait(epfd, &events, maxevents, timeout);
sigprocmask(SIG_SETMASK, &origmask, NULL);
```

The *sigmask* argument may be specified as NULL, in which case **epoll_pwait**() is equivalent to

epoll_wait().

RETURN VALUE

When successful, **epoll_wait**() returns the number of file descriptors ready for the requested I/O, or zero if no file descriptor became ready during the requested *timeout* milliseconds. When an error occurs, **epoll_wait**() returns −1 and *errno* is set appropriately.

ERRORS

EBADF

epfd is not a valid file descriptor.

EFAULT

The memory area pointed to by *events* is not accessible with write permissions.

EINTR

The call was interrupted by a signal handler before any of the requested events occurred or the *timeout* expired; see **signal**(7).

EINVAL

epfd is not an **epoll** file descriptor, or *maxevents* is less than or equal to zero.

VERSIONS

epoll_pwait() was added to Linux in kernel 2.6.19.

Glibc support for **epoll_pwait**() is provided starting with version 2.6.

CONFORMING TO

epoll_wait() is Linux-specific, and was introduced in kernel 2.5.44.

SEE ALSO

epoll_create(2), **epoll_ctl**(2), **epoll**(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/.