



Rocky Enterprise Linux 9.2 Manual Pages on command 'epoll_ctl.2'

C:\>man epoll_ctl.2

EPOLL_CTL(2) Linux Programmer's Manual EPOLL_CTL(2)

NAME

epoll_ctl - control interface for an epoll file descriptor

SYNOPSIS

```
#include <sys/epoll.h>
```

```
int epoll_ctl(int epfd, int op, int fd, struct epoll_event *event);
```

DESCRIPTION

This system call is used to add, modify, or remove entries in the interest list of the `epoll(7)` instance referred to by the file descriptor `epfd`. It requests that the operation `op` be performed for the target file descriptor, `fd`.

Valid values for the `op` argument are:

EPOLL_CTL_ADD

Add `fd` to the interest list and associate the settings specified in `event` with the internal file linked to `fd`.

EPOLL_CTL_MOD

Change the settings associated with `fd` in the interest list to the new settings specified in `event`.

EPOLL_CTL_DEL

Remove (deregister) the target file descriptor `fd` from the interest list.

The `event` argument is ignored and can be `NULL` (but see **BUGS** below).

The `event` argument describes the object linked to the file descriptor `fd`. 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 events member is a bit mask composed by ORing together zero or more of the following available event types:

lowing available event types:

EPOLLIN

The associated file is available for read(2) operations.

EPOLLOUT

The associated file is available for write(2) operations.

EPOLLRDHUP (since Linux 2.6.17)

Stream socket peer closed connection, or shut down writing half of connection. (This flag is especially useful for writing simple code to detect peer shutdown when using Edge Triggered monitoring.)

EPOLLPRI

There is an exceptional condition on the file descriptor. See the discussion of POLLPRI in poll(2).

EPOLLERR

Error condition happened on the associated file descriptor. This event is also reported for the write end of a pipe when the read end has been closed. `epoll_wait(2)` will always report for this event; it is not necessary to set it in events.

EPOLLHUP

Hang up happened on the associated file descriptor. `epoll_wait(2)` will always wait for this event; it is not necessary to set it in events.

Note that when reading from a channel such as a pipe or a stream socket, this event merely indicates that the peer closed its end of the channel.

Subsequent reads from the channel will return 0 (end of file) only after all outstanding data in the channel has been consumed.

EPOLLET

Sets the Edge Triggered behavior for the associated file descriptor. The default behavior for epoll is Level Triggered. See `epoll(7)` for more detailed information about Edge and Level Triggered event distribution architectures.

EPOLLONESHOT (since Linux 2.6.2)

Sets the one-shot behavior for the associated file descriptor. This means that after an event is pulled out with `epoll_wait(2)` the associated file descriptor is internally disabled and no other events will be reported by the epoll interface. The user must call `epoll_ctl()` with `EPOLL_CTL_MOD` to rearm the file descriptor with a new event mask.

EPOLLWAKEUP (since Linux 3.5)

If `EPOLLONESHOT` and `EPOLLET` are clear and the process has the `CAP_BLOCK_SUSPEND` capability, ensure that the system does not enter "suspend" or "hibernate" while this event is pending or being processed. The event is considered as being "processed" from the time when it is returned by a call to `epoll_wait(2)` until the next call to `epoll_wait(2)` on the same `epoll(7)` file descriptor, the closure of that file descriptor, the removal of the event file descriptor with `EPOLL_CTL_DEL`, or the clearing of `EPOLLWAKEUP` for the event file descriptor with `EPOLL_CTL_MOD`. See also `BUGS`.

EPOLLEXCLUSIVE (since Linux 4.5)

Sets an exclusive wakeup mode for the epoll file descriptor that is being attached to the target file descriptor, `fd`. When a wakeup event occurs and multiple epoll file descriptors are attached to the same target file using `EPOLLEXCLUSIVE`, one or more of the epoll file descriptors will receive an event with `epoll_wait(2)`. The default in this scenario (when `EPOLLEXCLUSIVE` is not set) is for all epoll file descriptors to receive an event. `EPOLLEXCLUSIVE` is thus useful for avoiding thundering herd problems in certain scenarios.

If the same file descriptor is in multiple epoll instances, some with the `EPOLLEXCLUSIVE` flag, and others without, then events will be provided to all

epoll instances that did not specify EPOLLEXCLUSIVE, and at least one of the epoll instances that did specify EPOLLEXCLUSIVE.

The following values may be specified in conjunction with EPOLLEXCLUSIVE:

EPOLLIN, EPOLLOUT, EPOLLWAKEUP, and EPOLLET. EPOLLHUP and EPOLLERR can also be specified, but this is not required: as usual, these events are always reported if they occur, regardless of whether they are specified in events.

Attempts to specify other values in events yield the error EINVAL.

EPOLLEXCLUSIVE may be used only in an EPOLL_CTL_ADD operation; attempts to employ it with EPOLL_CTL_MOD yield an error. If EPOLLEXCLUSIVE has been set using `epoll_ctl()`, then a subsequent EPOLL_CTL_MOD on the same `epfd`, `fd` pair yields an error. A call to `epoll_ctl()` that specifies EPOLLEXCLUSIVE in events and specifies the target file descriptor `fd` as an epoll instance will likewise fail. The error in all of these cases is EINVAL.

RETURN VALUE

When successful, `epoll_ctl()` returns zero. When an error occurs, `epoll_ctl()` returns -1 and `errno` is set appropriately.

ERRORS

EBADF `epfd` or `fd` is not a valid file descriptor.

EEXIST `op` was EPOLL_CTL_ADD, and the supplied file descriptor `fd` is already registered with this epoll instance.

EINVAL `epfd` is not an epoll file descriptor, or `fd` is the same as `epfd`, or the requested operation `op` is not supported by this interface.

EINVAL An invalid event type was specified along with EPOLLEXCLUSIVE in events.

EINVAL `op` was EPOLL_CTL_MOD and events included EPOLLEXCLUSIVE.

EINVAL `op` was EPOLL_CTL_MOD and the EPOLLEXCLUSIVE flag has previously been applied to this `epfd`, `fd` pair.

EINVAL EPOLLEXCLUSIVE was specified in event and `fd` refers to an epoll instance.

ELOOP `fd` refers to an epoll instance and this EPOLL_CTL_ADD operation would result in a circular loop of epoll instances monitoring one another.

ENOENT `op` was EPOLL_CTL_MOD or EPOLL_CTL_DEL, and `fd` is not registered with this epoll instance.

ENOMEM There was insufficient memory to handle the requested `op` control operation.

ENOSPC The limit imposed by `/proc/sys/fs/epoll/max_user_watches` was encountered

while trying to register (EPOLL_CTL_ADD) a new file descriptor on an epoll instance. See `epoll(7)` for further details.

EPERM The target file fd does not support epoll. This error can occur if fd refers to, for example, a regular file or a directory.

VERSIONS

`epoll_ctl()` was added to the kernel in version 2.6.

CONFORMING TO

`epoll_ctl()` is Linux-specific. Library support is provided in glibc starting with version 2.3.2.

NOTES

The epoll interface supports all file descriptors that support `poll(2)`.

BUGS

In kernel versions before 2.6.9, the EPOLL_CTL_DEL operation required a non-null pointer in event, even though this argument is ignored. Since Linux 2.6.9, event can be specified as NULL when using EPOLL_CTL_DEL. Applications that need to be portable to kernels before 2.6.9 should specify a non-null pointer in event.

If EPOLLWAKEUP is specified in flags, but the caller does not have the CAP_BLOCK_SUSPEND capability, then the EPOLLWAKEUP flag is silently ignored. This unfortunate behavior is necessary because no validity checks were performed on the flags argument in the original implementation, and the addition of the EPOLLWAKEUP with a check that caused the call to fail if the caller did not have the CAP_BLOCK_SUSPEND capability caused a breakage in at least one existing user-space application that happened to randomly (and uselessly) specify this bit. A robust application should therefore double check that it has the CAP_BLOCK_SUSPEND capability if attempting to use the EPOLLWAKEUP flag.

SEE ALSO

`epoll_create(2)`, `epoll_wait(2)`, `poll(2)`, `epoll(7)`

COLOPHON

This page is part of release 5.05 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at <https://www.kernel.org/doc/man-pages/>.