MyWebUniversity
python
Red Hat Enterprise Linux Manual Pages
FPDF Library PDF generator
FPDF
Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'epoll_ctl.2' command

$ 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 in?

terest 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 an entry to the interest list of the epoll file descriptor,

epfd. The entry includes the file descriptor, fd, a reference

to the corresponding open file description (see epoll(7) and

open(2)), and the settings specified in event.

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 in?

terest 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 data member of the epoll_event structure specifies data that the

kernel should save and then return (via epoll_wait(2)) when this file

descriptor becomes ready.

The events member of the epoll_event structure is a bit mask composed

by ORing together zero or more of the following 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 sim?

ple code to detect peer shutdown when using edge-triggered moni?

toring.)

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 nec?
essary to set it in events when calling epoll_ctl().

EPOLLHUP

Hang up happened on the associated file descriptor.

epoll_wait(2) will always wait for this event; it is not neces?
sary to set it in events when calling epoll_ctl().

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

Requests edge-triggered notification for the associated file de?
scriptor. The default behavior for epoll is level-triggered.
See epoll(7) for more detailed information about edge-triggered
and level-triggered notification.

This flag is an input flag for the event.events field when call?
ing epoll_ctl(); it is never returned by epoll_wait(2).

EPOLLONESHOT (since Linux 2.6.2)

Requests one-shot notification for the associated file descrip?
tor. This means that after an event notified for the file de?
scriptor by epoll_wait(2), the file descriptor is disabled in
the interest list 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.

This flag is an input flag for the event.events field when call?
ing epoll_ctl(); it is never returned by epoll_wait(2).

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) un?
til the next call to epoll_wait(2) on the same epoll(7) file de?
scriptor, 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.

This flag is an input flag for the event.events field when call?
ing epoll_ctl(); it is never returned by epoll_wait(2).

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 at?
tached 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 EPOLLEXCLU?
SIVE 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 re?
quired: as usual, these events are always reported if they oc?
cur, regardless of whether they are specified in events. At?
tempts 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 spec?
ifies the target file descriptor fd as an epoll instance will
likewise fail. The error in all of these cases is EINVAL.

The EPOLLEXCLUSIVE flag is an input flag for the event.events
field   when calling epoll_ctl(); it is never returned by
epoll_wait(2).

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 al?
ready 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 inter?
face.

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 or a nesting depth of epoll instances greater than
5.

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 en?

countered while trying to register (EPOLL_CTL_ADD) a new file
descriptor on an epoll instance. See epoll(7) for further de?
tails.

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. Library support is
provided in glibc starting with version 2.3.2.

CONFORMING TO

epoll_ctl() is Linux-specific.

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 ig?
nored. This unfortunate behavior is necessary because no validity
checks were performed on the flags argument in the original implementa?
tion, 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 capabil?
ity 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 5.10 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/.