NAME

ioctl_eventpoll, EPIOCSPARAMS, EPIOCGPARAMS − ioctl() operations for epoll file descriptors

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <sys/epoll.h> /* Definition of **EPIOC*** constants */
#include <sys/ioctl.h>

int ioctl(int *fd***, EPIOCSPARAMS, const struct epoll_params ****argp***);**
int ioctl(int *fd***, EPIOCGPARAMS, struct epoll_params ****argp***);**

#include <sys/epoll.h>

```
struct epoll_params {
    uint32_t  busy_poll_usecs;  /* Number of usecs to busy poll */
    uint16_t  busy_poll_budget;  /* Max packets per poll */
    uint8_t   prefer_busy_poll;/* Boolean preference  */

    /* pad the struct to a multiple of 64bits */
    uint8_t   __pad;            /* Must be zero */
};
```

DESCRIPTION

EPIOCSPARAMS
Set the *epoll_params* structure to configure the operation of epoll. Refer to the structure description below to learn what configuration is supported.

EPIOCGPARAMS
Get the current *epoll_params* configuration settings.

All operations documented above must be performed on an epoll file descriptor, which can be obtained with a call to **epoll_create**(2) or **epoll_create1**(2).

The epoll_params structure

argp.busy_poll_usecs denotes the number of microseconds that the network stack will busy poll. During this time period, the network device will be polled repeatedly for packets. This value cannot exceed **INT_MAX**.

argp.busy_poll_budget denotes the maximum number of packets that the network stack will retrieve on each poll attempt. This value cannot exceed **NAPI_POLL_WEIGHT** (which is 64 as of Linux 6.9), unless the process is run with **CAP_NET_ADMIN**.

argp.prefer_busy_poll is a boolean field and must be either 0 (disabled) or 1 (enabled). If enabled, this indicates to the network stack that busy poll is the preferred method of processing network data and the network stack should give the application the opportunity to busy poll. Without this option, very busy systems may continue to do network processing via the normal method of IRQs triggering softIRQ and NAPI.

argp.__pad must be zero.

RETURN VALUE

On success, 0 is returned. On failure, −1 is returned, and *errno* is set to indicate the error.

ERRORS

EOPNOTSUPP
The kernel was not compiled with busy poll support.

EINVAL
fd is not a valid file descriptor.

EINVAL
argp.__pad is not zero.

EINVAL
argp.busy_poll_usecs exceeds **INT_MAX**.

EINVAL
argp.prefer_busy_poll is not 0 or 1.

EPERM
The process is being run without **CAP_NET_ADMIN** and the specified *argp.busy_poll_budget* exceeds **NAPI_POLL_WEIGHT**.

EFAULT
argp is an invalid address.

STANDARDS
Linux.

HISTORY
Linux 6.9, glibc 2.40.

EXAMPLES
```
/* Code to set the epoll params to enable busy polling */

int epollfd = epoll_create1(0);
struct epoll_params params;

if (epollfd == -1) {
    perror("epoll_create1");
    exit(EXIT_FAILURE);
}

memset(&params, 0, sizeof(struct epoll_params));

params.busy_poll_usecs = 25;
params.busy_poll_budget = 8;
params.prefer_busy_poll = 1;

if (ioctl(epollfd, EPIOCSPARAMS, &params) == -1) {
    perror("ioctl");
    exit(EXIT_FAILURE);
}

/* Code to show how to retrieve the current settings */

memset(&params, 0, sizeof(struct epoll_params));

if (ioctl(epollfd, EPIOCGPARAMS, &params) == -1) {
    perror("ioctl");
    exit(EXIT_FAILURE);
}

/* params struct now contains the current parameters */

fprintf(stderr, "epoll usecs: %lu\n", params.busy_poll_usecs);
fprintf(stderr, "epoll packet budget: %u\n", params.busy_poll_budget);
fprintf(stderr, "epoll prefer busy poll: %u\n", params.prefer_busy_poll);
```

SEE ALSO

ioctl(2), **epoll_create**(2), **epoll_create1**(2), **epoll**(7)

linux.git/Documentation/networking/napi.rst

linux.git/Documentation/admin-guide/sysctl/net.rst