## NAME

**cpuset_getdomain**, **cpuset_setdomain** - manage memory domain policy

## LIBRARY

Standard C Library (libc, -lc)

## SYNOPSIS

**#include <sys/param.h>**
**#include <sys/domainset.h>**

*int*
**cpuset_getdomain**(*cpulevel_t level, cpuwhich_t which, id_t id, size_t setsize, domainset_t *mask, int *policy*);

*int*
**cpuset_setdomain**(*cpulevel_t level, cpuwhich_t which, id_t id, size_t setsize, const domainset_t *mask, int policy*);

## DESCRIPTION

**cpuset_getdomain**() and **cpuset_setdomain**() allow the manipulation of sets of memory domains and allocation policy available to processes, threads, jails and other resources. These functions may manipulate sets of memory domains that contain many processes or per-object anonymous masks that affect only a single object.

The valid values for the *level* and *which* arguments are documented in cpuset(2). These arguments specify which object and which set of the object we are referring to. Not all possible combinations are valid. For example, only processes may belong to a numbered set accessed by a *level* argument of CPU_LEVEL_CPUSET. All resources, however, have a mask which may be manipulated with CPU_LEVEL_WHICH.

Masks of type *domainset_t* are composed using the DOMAINSET macros. The kernel tolerates large sets as long as all domains specified in the set exist. Sets smaller than the kernel uses generate an error on calls to **cpuset_getdomain**() even if the result set would fit within the user supplied set. Calls to **cpuset_setdomain**() tolerate small sets with no restrictions.

The supplied mask should have a size of *setsize* bytes. This size is usually provided by calling sizeof(mask) which is ultimately determined by the value of DOMAINSET_SETSIZE as defined in <*sys/domainset.h*>.

**cpuset_getdomain**() retrieves the mask and policy from the object specified by *level*, *which* and *id* and

stores it in the space provided by *mask* and *policy*.

**cpuset_setdomain**() attempts to set the mask and policy for the object specified by *level*, *which* and *id* to the values in *mask* and *policy*.

## ALLOCATION POLICIES

Valid policy values are as follows:

DOMAINSET_POLICY_ROUNDROBIN
: Memory is allocated on a round-robin basis by cycling through each domain in *mask*.

DOMAINSET_POLICY_FIRSTTOUCH
: Memory is allocated on the domain local to the CPU the requesting thread is running on. Failure to allocate from this domain will fallback to round-robin.

DOMAINSET_POLICY_PREFER
: Memory is allocated preferentially from the single domain specified in the mask. If memory is unavailable the domains listed in the parent cpuset will be visited in a round-robin order.

## RETURN VALUES

Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

## ERRORS

The following error codes may be set in *errno*:

[EINVAL] The *level* or *which* argument was not a valid value.

[EINVAL] The *mask or policy* argument specified when calling **cpuset_setdomain**() was not a valid value.

[EDEADLK] The **cpuset_setdomain**() call would leave a thread without a valid CPU to run on because the set does not overlap with the thread’s anonymous mask.

[EFAULT] The mask pointer passed was invalid.

[ESRCH] The object specified by the *id* and *which* arguments could not be found.

[ERANGE] The *domainsetsize* was either preposterously large or smaller than the kernel set size.

[EPERM] The calling process did not have the credentials required to complete the operation.

[ECAPMODE] The calling process attempted to act on a process other than itself, while in capability mode. See capsicum(4).

## SEE ALSO

cpuset(1), cpuset(2), cpuset_getaffinity(2), cpuset_getid(2), cpuset_setaffinity(2), cpuset_setid(2), capsicum(4), cpuset(9)

## HISTORY

The **cpuset_getdomain** family of system calls first appeared in FreeBSD 12.0.

## AUTHORS

Jeffrey Roberson <*jeff@FreeBSD.org*>