NAME

cpuset, **cpuset_getid**, **cpuset_setid** - manage CPU affinity sets

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/param.h>
#include <sys/cpuset.h>

int
cpuset(*cpusetid_t *setid*);

int
cpuset_setid(*cpuwhich_t which*, *id_t id*, *cpusetid_t setid*);

int
cpuset_getid(*cpulevel_t level*, *cpuwhich_t which*, *id_t id*, *cpusetid_t *setid*);

DESCRIPTION

The **cpuset** family of system calls allow applications to control sets of processors and memory domains and assign processes and threads to these sets. Processor sets contain lists of CPUs and domains that members may run on and exist only as long as some process is a member of the set. All processes in the system have an assigned set. The default set for all processes in the system is the set numbered 1. Threads belong to the same set as the process which contains them, however, they may further restrict their set with the anonymous per-thread mask to bind to a specific CPU or subset of CPUs and memory domains.

Sets are referenced by a number of type *cpuset_id_t*. Each thread has a root set, an assigned set, and an anonymous mask. Only the root and assigned sets are numbered. The root set is the set of all CPUs and memory domains available in the system or in the system partition the thread is running in. The assigned set is a subset of the root set and is administratively assignable on a per-process basis. Many processes and threads may be members of a numbered set.

The anonymous set is a further thread-specific refinement on the assigned set. It is intended that administrators will manipulate numbered sets using cpuset(1) while application developers will manipulate anonymous sets using cpuset_setaffinity(2) and cpuset_setdomain(2).

To select the correct set a value of type *cpulevel_t* is used. The following values for *level* are supported:

CPU_LEVEL_ROOT	Root set
CPU_LEVEL_CPUSET	Assigned set
CPU_LEVEL_WHICH	Set specified by which argument

The *which* argument determines how the value of *id* is interpreted and is of type *cpuwhich_t*. The *which* argument may have the following values:

CPU_WHICH_TID	id is lwpid_t (thread id)
CPU_WHICH_PID	id is pid_t (process id)
CPU_WHICH_TIDPID	id is either a thread or process id
CPU_WHICH_JAIL	id is jid (jail id)
CPU_WHICH_CPUSET	id is a cpusetid_t (cpuset id)
CPU_WHICH_IRQ	id is an irq number
CPU_WHICH_INTRHANDLER	id is an irq number for an interrupt handler
CPU_WHICH_ITHREAD	id is an irq number for an ithread
CPU_WHICH_DOMAIN	id is a NUMA domain

An *id* of '-1' may be used with a *which* of CPU_WHICH_TID, CPU_WHICH_PID, CPU_WHICH_TIDPID, or CPU_WHICH_CPUSET to mean the current thread, process, or current thread's cpuset. All cpuset syscalls allow this usage.

A *level* argument of CPU_LEVEL_WHICH combined with a *which* argument other than CPU_WHICH_CPUSET refers to the anonymous mask of the object. This mask does not have an id and may only be manipulated with cpuset_setaffinity(2).

cpuset() creates a new set containing the same CPUs as the root set of the current process and stores its id in the space provided by *setid*. On successful completion the calling process joins the set and is the only member. Children inherit this set after a call to fork(2).

cpuset_setid() attempts to set the id of the object specified by the *which* argument. Currently CPU_WHICH_PID is the only acceptable value for which as threads do not have an id distinct from their process and the API does not permit changing the id of an existing set. Upon successful completion all of the threads in the target process will be running on CPUs permitted by the set.

cpuset_getid() retrieves a set id from the object indicated by *which* and stores it in the space pointed to by *setid*. The retrieved id may be that of either the root or assigned set depending on the value of *level*. *level* should be CPU_LEVEL_CPUSET or CPU_LEVEL_ROOT to get the set id from the process or thread specified by the *id* argument. Specifying CPU_LEVEL_WHICH with a process or thread is unsupported since this references the unnumbered anonymous mask.

The actual contents of the sets may be retrieved or manipulated using cpuset_getaffinity(2), cpuset_setaffinity(2), cpuset_getdomain(2), and cpuset_setdomain(2). The cpuset(9) macros may be used to manipulate masks of type *cpuset_t* get and set using those APIs. See those manual pages for more detail.

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.

EXAMPLES

In this example, a CPU set mask is configured to limit execution to the first CPU using CPU_ZERO(9) and CPU_SET(9), members of the cpuset(9) programming interface. Then, the mask is applied to a new anonymous CPU set associated with the current process using cpuset_setaffinity(2). This mask will be used by the current process, and inherited by any new child processes.

```
#include <sys/param.h>
#include <sys/cpuset.h>

#include <sysexits.h>

cpuset_t cpuset_mask;

/* Initialize a CPU mask and enable CPU 0. */
CPU_ZERO(&cpuset_mask);
CPU_SET(0, &cpuset_mask);

/* Set affinity for the CPU set for the current process. */
if (cpuset_setaffinity(CPU_LEVEL_WHICH, CPU_WHICH_PID, -1,
    sizeof(cpuset_mask), &cpuset_mask) < 0)
        err(EX_OSERR, "cpuset_setaffinity");
```

In the next example, a named CPU set is created containing the current process, and its affinity similarly configured. The resulting CPU set ID can then be used for further external management of the affinity of the set.

```
#include <sys/param.h>
#include <sys/cpuset.h>

#include <sysexits.h>
```

```
cpusetid_t cpuset_id;
cpuset_t cpuset_mask;

/* Create new cpuset for the current process. */
if (cpuset(&cpuset_id) < 0)
        err(EX_OSERR, "cpuset");

/* Initialize a CPU mask and enable CPU 0. */
CPU_ZERO(&cpuset_mask);
CPU_SET(0, &cpuset_mask);

/* Set affinity for the CPU set for the current process. */
if (cpuset_setaffinity(CPU_LEVEL_SET, CPU_WHICH_CPUSET, cpuset_id,
   sizeof(cpuset_mask), &cpuset_mask) < 0)
        err(EX_OSERR, "cpuset_setaffinity");
```

ERRORS

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

[EINVAL]	The *which* or *level* argument was not a valid value.
[EDEADLK]	The **cpuset_setid**() 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 setid pointer passed to **cpuset_getid**() or **cpuset**() was invalid.
[ESRCH]	The object specified by the *id* and *which* arguments could not be found.
[EPERM]	The calling process did not have the credentials required to complete the operation.
[ENFILE]	There was no free *cpusetid_t* for allocation.

SEE ALSO

cpuset(1), cpuset_getaffinity(2), cpuset_getdomain(2), cpuset_setaffinity(2), cpuset_setdomain(2), pthread_affinity_np(3), pthread_attr_affinity_np(3), CPU_SET(9), CPU_ZERO(9), cpuset(9)

HISTORY

The **cpuset** family of system calls first appeared in FreeBSD 7.1.

AUTHORS

Jeffrey Roberson <*jeff@FreeBSD.org*>