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

$ man spu_create.2

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

NAME

spu_create - create a new spu context

SYNOPSIS

#include <sys/types.h>

#include <sys/spu.h>

int spu_create(const char *pathname, int flags, mode_t mode,

int neighbor_fd);

Note: There is no glibc wrapper for this system call; see NOTES.

DESCRIPTION

The spu_create() system call is used on PowerPC machines that implement the Cell Broadband Engine Architecture in order to access Synergistic Processor Units (SPUs). It creates a new logical context for an SPU in pathname and returns a file descriptor associated with it. pathname must refer to a nonexistent directory in the mount point of the SPU filesystem (spufs). If spu_create() is successful, a directory is cre? ated at pathname and it is populated with the files described in spufs(7).

When a context is created, the returned file descriptor can only be passed to spu_run(2), used as the dirfd argument to the *at family of system calls (e.g., openat(2)), or closed; other operations are not de? fined. A logical SPU context is destroyed (along with all files cre? ated within the context's pathname directory) once the last reference to the context has gone; this usually occurs when the file descriptor returned by spu_create() is closed.

The mode argument (minus any bits set in the process's umask(2)) speci? fies the permissions used for creating the new directory in spufs. See stat(2) for a full list of the possible mode values.

The neighbor_fd is used only when the SPU_CREATE_AFFINITY_SPU flag is specified; see below.

The flags argument can be zero or any bitwise OR-ed combination of the following constants:

SPU_CREATE_EVENTS_ENABLED

Rather than using signals for reporting DMA errors, use the event argument to spu_run(2).

SPU_CREATE_GANG

Create an SPU gang instead of a context. (A gang is a group of SPU contexts that are functionally related to each other and which share common scheduling parameters?priority and policy. In the future, gang scheduling may be implemented causing the group to be switched in and out as a single unit.)

A new directory will be created at the location specified by the pathname argument. This gang may be used to hold other SPU con? texts, by providing a pathname that is within the gang directory to further calls to spu_create().

SPU_CREATE_NOSCHED

Create a context that is not affected by the SPU scheduler. Once the context is run, it will not be scheduled out until it is destroyed by the creating process.

Because the context cannot be removed from the SPU, some func? tionality is disabled for SPU_CREATE_NOSCHED contexts. Only a

subset of the files will be available in this context directory in spufs. Additionally, SPU_CREATE_NOSCHED contexts cannot dump a core file when crashing.

Creating SPU_CREATE_NOSCHED contexts requires the CAP_SYS_NICE capability.

SPU_CREATE_ISOLATE

Create an isolated SPU context. Isolated contexts are protected from some PPE (PowerPC Processing Element) operations, such as access to the SPU local store and the NPC register.

Creating SPU_CREATE_ISOLATE contexts also requires the SPU_CRE?ATE_NOSCHED flag.

SPU_CREATE_AFFINITY_SPU (since Linux 2.6.23)

Create a context with affinity to another SPU context. This affinity information is used within the SPU scheduling algo?rithm. Using this flag requires that a file descriptor refer?ring to the other SPU context be passed in the neighbor_fd argu?ment.

SPU_CREATE_AFFINITY_MEM (since Linux 2.6.23)

Create a context with affinity to system memory. This affinity information is used within the SPU scheduling algorithm.

RETURN VALUE

On success, spu_create() returns a new file descriptor. On error, -1 is returned, and errno is set to one of the error codes listed below.

ERRORS

EACCES The current user does not have write access to the spufs(7) mount point.

EEXIST An SPU context already exists at the given pathname.

EFAULT pathname is not a valid string pointer in the calling process's address space.

EINVAL pathname is not a directory in the spufs(7) mount point, or in?valid flags have been provided.

ELOOP Too many symbolic links were found while resolving pathname.

EMFILE The per-process limit on the number of open file descriptors has

been reached.

ENAMETOOLONG

pathname is too long.

ENFILE The system-wide limit on the total number of open files has been reached.

ENODEV An isolated context was requested, but the hardware does not support SPU isolation.

ENOENT Part of pathname could not be resolved.

ENOMEM The kernel could not allocate all resources required.

ENOSPC There are not enough SPU resources available to create a new context or the user-specific limit for the number of SPU con? texts has been reached.

ENOSYS The functionality is not provided by the current system, because either the hardware does not provide SPUs or the spufs module is not loaded.

ENOTDIR

A part of pathname is not a directory.

EPERM The SPU_CREATE_NOSCHED flag has been given, but the user does not have the CAP_SYS_NICE capability.

FILES

pathname must point to a location beneath the mount point of spufs. By convention, it gets mounted in /spu.

VERSIONS

The spu_create() system call was added to Linux in kernel 2.6.16.

CONFORMING TO

This call is Linux-specific and implemented only on the PowerPC archi? tecture. Programs using this system call are not portable.

NOTES

Glibc does not provide a wrapper for this system call; call it using syscall(2). Note however, that spu_create() is meant to be used from libraries that implement a more abstract interface to SPUs, not to be used from regular applications. See ?http://www.bsc.es/projects /deepcomputing/linuxoncell/? for the recommended libraries.

Prior to the addition of the SPU_CREATE_AFFINITY_SPU flag in Linux 2.6.23, the spu_create() system call took only three arguments (i.e., there was no neighbor_fd argument).

EXAMPLES

See spu_run(2) for an example of the use of spu_create()

SEE ALSO

close(2), spu_run(2), capabilities(7), spufs(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/.