NAME

pmc_capabilities, **pmc_cpuinfo**, **pmc_ncpu**, **pmc_npmc**, **pmc_pmcinfo**, **pmc_width** - retrieve information about performance monitoring counters

LIBRARY

Performance Counters Library (libpmc, -lpmc)

SYNOPSIS

#include <pmc.h>

int
pmc_capabilities(*pmc_id_t pmc, uint32_t *caps*);

int
pmc_cpuinfo(*const struct pmc_cpuinfo **cpu_info*);

int
pmc_ncpu(*void*);

int
pmc_npmc(*int cpu*);

int
pmc_pmcinfo(*int cpu, struct pmc_pmcinfo **pmc_info*);

int
pmc_width(*pmc_id_t pmc, uint32_t *width*);

DESCRIPTION

These functions retrieve information about performance monitoring hardware.

Function **pmc_capabilities**() retrieves the hardware capabilities of a PMC. Argument *pmc* is a PMC handle obtained by a prior call to **pmc_allocate**(). The function sets argument *caps* to a bit mask of capabilities supported by the PMC denoted by argument *pmc*. PMC capabilities are described in pmc(3).

Function **pmc_cpuinfo**() retrieves information about the CPUs in the system. Argument *cpu_info* will be set to point to an internal structure with information about the system's CPUs. The caller should not free this pointer value. This structure has the following fields:

pm_cputype

Specifies the CPU type.

pm_ncpu Specifies the number of CPUs in the system.

pm_npmc Specifies the number of PMC rows per CPU.

pm_nclass Specifies the number of distinct classes of PMCs in the system.

pm_classes Contains an array of *struct pmc_classinfo* descriptors describing the properties of each class of PMCs in the system.

Function **pmc_ncpu**() is a convenience function that returns the maximum CPU number in the system. On systems that support sparsely numbered CPUs, not all CPUs may be physically present. Applications need to be prepared to deal with nonexistent CPUs.

Function **pmc_npmc**() is a convenience function that returns the number of PMCs available in the CPU specified by argument *cpu*.

Function **pmc_pmcinfo**() returns information about the current state of the PMC hardware in the CPU specified by argument *cpu*. The location specified by argument *pmc_info* is set to point an array of *struct pmc_info* structures each describing the state of one PMC in the CPU. These structure contain the following fields:

pm_name A human readable name for the PMC.

pm_class The PMC class for the PMC.

pm_enabled Non-zero if the PMC is enabled.

pm_rowdisp The disposition of the PMC row for this PMC. Row dispositions are documented in hwpmc(4).

pm_ownerpid If the hardware is in use, the process id of the owner of the PMC.

pm_mode The PMC mode as described in pmc(3).

pm_event If the hardware is in use, the PMC event being measured.

pm_flags If the hardware is in use, the flags associated with the PMC.

pm_reloadcount For sampling PMCs, the reload count associated with the PMC.

Function **pmc_width**() is used to retrieve the width in bits of the hardware counters associated with a PMC. Argument *pmc* is a PMC handle obtained by a prior call to **pmc_allocate**(). The function sets the location pointed to by argument *width* to the width of the physical counters associated with PMC *pmc*.

RETURN VALUES

Functions **pmc_ncpu**() and **pmc_npmc**() returns a positive integer if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

Functions **pmc_capabilities**(), **pmc_cpuinfo**(), **pmc_pmcinfo**() and **pmc_width**() return 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

A call to function **pmc_capabilities**() may fail with the following errors:

[EINVAL] The argument to the function was invalid.

Calls to functions **pmc_cpuinfo**(), **pmc_ncpu**() and **pmc_npmc**() may fail with the following errors:

[ENXIO] A prior call to **pmc_init**() to initialize the PMC library had failed.

A call to function **pmc_pmcinfo**() may fail with the following errors:

[EINVAL] The argument *cpu* was invalid.

[ENXIO] The argument *cpu* specified a disabled or absent CPU.

A call to function **pmc_width**() may fail with the following errors:

[EINVAL] The argument to the function was invalid.

SEE ALSO

pmc(3), pmc_allocate(3), pmc_get_driver_stats(3), pmc_name_of_capability(3), pmc_name_of_class(3), pmc_name_of_cputype(3), pmc_name_of_event(3), pmc_name_of_mode(3), hwpmc(4)