NAME

pmc_allocate, **pmc_release** - allocate and free performance monitoring counters

LIBRARY

Performance Counters Library (libpmc, -lpmc)

SYNOPSIS

#include <pmc.h>

int
pmc_allocate(*const char *eventspecifier*, *enum pmc_mode mode*, *uint32_t flags*, *int cpu*,
*pmc_id_t *pmcid*, *uint64_t count*);

int
pmc_release(*pmc_id_t pmc*);

DESCRIPTION

Function **pmc_allocate**() allocates a performance monitoring counter that measures the events named by argument *eventspecifier*, and writes the allocated handle to the location pointed to by argument *pmcid*.

Argument *eventspecifier* comprises an PMC event name followed by an optional comma separated list of keywords and qualifiers. The allowed syntax for argument *eventspecifier* is processor specific and is listed in section *EVENT SPECIFIERS* in the pmc(3) manual page.

The desired PMC mode is specified by argument *mode*. Legal values for the *mode* argument are:

PMC_MODE_SC	Allocate a system-scope counting PMC.
PMC_MODE_SS	Allocate a system-scope sampling PMC.
PMC_MODE_TC	Allocate a process-scope counting PMC.
PMC_MODE_TS	Allocate a process-scope sampling PMC.

Mode specific modifiers may be specified using argument *flags*. The flags supported at PMC allocation time are:

PMC_F_DESCENDANTS	For process-scope PMCs, automatically track descendants of attached processes.
PMC_F_LOG_PROCCSW	For process-scope counting PMCs, generate a log event at every context switch containing the incremental number of hardware events seen by the process during the time it was executing on the CPU.

PMC_F_LOG_PROCEXIT	For process-scope counting PMCs, accumulate hardware events seen when the process was executing on a CPU and generate a log event when an attached process exits.

PMCs allocated with flags PMC_F_LOG_PROCCSW and PMC_F_LOG_PROCEXIT need a log file to be configured before they are started.

For system scope PMCs, the argument *cpu* is a non-negative value that specifies the CPU number that the PMC is to be allocated on. Process scope PMC allocations should specify the constant PMC_CPU_ANY for this argument.

The *count* argument behaves identically to the pmc_set(3) function’s *value* argument. For counting PMCs, *count* specifies the initial value of the allocated PMC. For sampling PMCs, *count* specifies the reload count.

Function **pmc_release**() releases the PMC denoted by argument *pmcid*.

RETURN VALUES

If successful, function **pmc_allocate**() sets the location specified by argument *pmcid* to the handle of the allocated PMC and returns 0. In case of an error, the function returns -1 and sets the global variable *errno* to indicate the error.

The **pmc_release**() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

[EINVAL]	The argument *mode* to function **pmc_allocate**() had an invalid value.
[EINVAL]	Argument *cpu* to function **pmc_allocate**() had an invalid CPU number.
[EINVAL]	Argument *flags* contained flags that were unsupported or otherwise incompatible with the requested PMC mode.
[EINVAL]	Argument *eventspecifier* to function **pmc_allocate**() specified an event not supported by hardware or contained a syntax error.
[ENXIO]	Function **pmc_allocate**() requested the use of a hardware resource that was absent or administratively disabled.
[EOPNOTSUPP]	The underlying hardware does not support the capabilities needed for a PMC being allocated by a call to **pmc_allocate**().

[EPERM]	A system scope PMC allocation was attempted without adequate process privilege.
[ESRCH]	Function **pmc_release**() was called without first having allocated a PMC.
[EINVAL]	Argument *pmcid* to function **pmc_release**() did not specify a PMC previously allocated by this process.

SEE ALSO

pmc(3), pmc_attach(3), pmc_configure_logfile(3), pmc_start(3), hwpmc(4)