NAME
       perfmonctl − interface to IA-64 performance monitoring unit

SYNOPSIS
       #include <syscall.h>
       #include <perfmon.h>

       long perfmonctl(int *fd***, int** *cmd***, void ****arg***, int** *narg***);**
       Note: There is no glibc wrapper for this system call; see NOTES.

DESCRIPTION
       The IA-64-specific **perfmonctl**() system call provides an interface to the PMU (performance monitoring
       unit). The PMU consists of PMD (performance monitoring data) registers and PMC (performance monitor-
       ing control) registers, which gather hardware statistics.

       perfmonctl() applies the operation *cmd* to the input arguments specified by *arg*. The number of arguments
       is defined by *narg*. The *fd* argument specifies the perfmon context to operate on.

       Supported values for *cmd* are:

PFM_CREATE_CONTEXT
       perfmonctl(int *fd***, PFM_CREATE_CONTEXT, pfarg_context_t ****ctxt***, 1);**
       Set up a context.

       The *fd* parameter is ignored. A new perfmon context is created as specified in *ctxt* and its file de-
       scriptor is returned in *ctxt->ctx_fd*.

       The file descriptor can be used in subsequent calls to **perfmonctl**() and can be used to read event
       notifications (type *pfm_msg_t*) using **read**(2). The file descriptor is pollable using **select**(2),
       poll(2), and **epoll**(7).

       The context can be destroyed by calling **close**(2) on the file descriptor.

PFM_WRITE_PMCS
       perfmonctl(int *fd***, PFM_WRITE_PMCS, pfarg_reg_t ****pmcs***, n);**
       Set PMC registers.

PFM_WRITE_PMDS
       perfmonctl(int *fd***, PFM_WRITE_PMDS, pfarg_reg_t ****pmds***, n);**
       Set PMD registers.

PFM_READ_PMDS
       perfmonctl(int *fd***, PFM_READ_PMDS, pfarg_reg_t ****pmds***, n);**
       Read PMD registers.

PFM_START
       perfmonctl(int *fd***, PFM_START, NULL, 0);**
       Start monitoring.

PFM_STOP
       perfmonctl(int *fd***, PFM_STOP, NULL, 0);**
       Stop monitoring.

PFM_LOAD_CONTEXT
       perfmonctl(int *fd***, PFM_LOAD_CONTEXT, pfarg_load_t ****largs***, 1);**
       Attach the context to a thread.

PFM_UNLOAD_CONTEXT
       perfmonctl(int *fd***, PFM_UNLOAD_CONTEXT, NULL, 0);**
       Detach the context from a thread.

PFM_RESTART
       perfmonctl(int *fd***, PFM_RESTART, NULL, 0);**
       Restart monitoring after receiving an overflow notification.

PFM_GET_FEATURES
> **perfmonctl(int** *fd***, PFM_GET_FEATURES, pfarg_features_t ****arg***, 1);**

PFM_DEBUG
> **perfmonctl(int** *fd***, PFM_DEBUG,** *val***, 0);**
> If *val* is nonzero, enable debugging mode, otherwise disable.

PFM_GET_PMC_RESET_VAL
> **perfmonctl(int** *fd***, PFM_GET_PMC_RESET_VAL, pfarg_reg_t ****req***, n);**
> Reset PMC registers to default values.

RETURN VALUE
perfmonctl() returns zero when the operation is successful. On error, −1 is returned and *errno* is set to indicate the cause of the error.

VERSIONS
perfmonctl() is available since Linux 2.4.

CONFORMING TO
perfmonctl() is Linux-specific and is available only on the IA-64 architecture.

NOTES
Glibc does not provide a wrapper for this system call; call it using **syscall**(2).

SEE ALSO
gprof(1)

The perfmon2 interface specification

COLOPHON
This page is part of release 5.02 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/.