



Linux Ubuntu 26.04 Manual Pages on command 'perfmonctl.2'

\$ man perfmonctl.2

perfmonctl(2) System Calls Manual perfmonctl(2)

NAME

perfmonctl - interface to IA-64 performance monitoring unit

SYNOPSIS

```
#include <syscall.h>
```

```
#include <perfmon.h>
```

```
long perfmonctl(int nargs;  
                int fd, int cmd, void arg[narg], int nargs);
```

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

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 monitoring 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 descriptor 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 *larges, 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 error.

STANDARDS

Linux on IA-64.

HISTORY

Added in Linux 2.4; removed in Linux 5.10.

This system call was broken for many years, and ultimately removed in Linux 5.10.

glibc does not provide a wrapper for this system call; on kernels where it exists, call it using syscall(2).

SEE ALSO

gprof(1)

The perfmon2 interface specification

Linux man-pages 6.17

2026-02-08

perfmonctl(2)