NAME

pmem_flush(), **pmem_drain**(), **pmem_persist**(), **pmem_msync**(), **pmem_deep_flush**(), **pmem_deep_drain**(), **pmem_deep_persist**(), **pmem_has_hw_drain**(), **pmem_has_auto_flush**() – check persistency, store persistent data and delete mappings

SYNOPSIS

```
#include <libpmem.h>

void pmem_persist(const void *addr, size_t len);
int pmem_msync(const void *addr, size_t len);
void pmem_flush(const void *addr, size_t len);
void pmem_deep_flush(const void *addr, size_t len); (EXPERIMENTAL)
int pmem_deep_drain(const void *addr, size_t len); (EXPERIMENTAL)
int pmem_deep_persist(const void *addr, size_t len); (EXPERIMENTAL)
void pmem_drain(void);
int pmem_has_auto_flush(void); (EXPERIMENTAL)
int pmem_has_hw_drain(void);
```

DESCRIPTION

The functions in this section provide access to the stages of flushing to persistence, for the less common cases where an application needs more control of the flushing operations than the **pmem_persist**() function.

WARNING: Using **pmem_persist**() on a range where **pmem_is_pmem**(3) returns false may not do anything useful – use **msync**(2) instead.

The **pmem_persist**() function force any changes in the range [*addr*, *addr*+*len*) to be stored durably in persistent memory. This is equivalent to calling **msync**(2) but may be more optimal and will avoid calling into the kernel if possible. There are no alignment restrictions on the range described by *addr* and *len*, but **pmem_persist**() may expand the range as necessary to meet platform alignment requirements.

WARNING: Like **msync**(2), there is nothing atomic or transactional about this call. Any unwritten stores in the given range will be written, but some stores may have already been written by virtue of normal cache eviction/replacement policies. Correctly written code must not depend on stores waiting until **pmem_persist**() is called to become persistent – they can become persistent at any time before **pmem_persist**() is called.

The **pmem_msync**() function is like **pmem_persist**() in that it forces any changes in the range [*addr*, *addr*+*len*) to be stored durably. Since it calls **msync**(), this function works on either persistent memory or a memory mapped file on traditional storage. **pmem_msync**() takes steps to ensure the alignment of addresses and lengths passed to **msync**() meet the requirements of that system call. It calls **msync**() with the **MS_SYNC** flag as described in **msync**(2). Typically the application only checks for the existence of persistent memory once, and then uses that result throughout the program, for example:

```
/* do this call once, after the pmem is memory mapped */
int is_pmem = pmem_is_pmem(rangeaddr, rangelen);

/* ... make changes to a range of pmem ... */

/* make the changes durable */
if (is_pmem)
    pmem_persist(subrangeaddr, subrangelen);
else
    pmem_msync(subrangeaddr, subrangelen);

/* ... */
```

WARNING: On Linux, **pmem_msync**() and **msync**(2) have no effect on memory ranges mapped from Device DAX. In case of memory ranges where **pmem_is_pmem**(3) returns true use

pmem_persist() to force the changes to be stored durably in persistent memory.

The **pmem_flush**() and **pmem_drain**() functions provide partial versions of the **pmem_persist**() function. **pmem_persist**() can be thought of as this:

```
void
pmem_persist(const void *addr, size_t len)
{
    /* flush the processor caches */
    pmem_flush(addr, len);

    /* wait for any pmem stores to drain from HW buffers */
    pmem_drain();
}
```

These functions allow advanced programs to create their own variations of **pmem_persist**(). For example, a program that needs to flush several discontiguous ranges can call **pmem_flush**() for each range and then follow up by calling **pmem_drain**() once.

The semantics of **pmem_deep_flush**() function is the same as **pmem_flush**() function except that **pmem_deep_flush**() is indifferent to **PMEM_NO_FLUSH** environment variable (see **ENVIRONMENT** section in **libpmem**(7)) and always flushes processor caches.

The behavior of **pmem_deep_persist**() function is the same as **pmem_persist**(), except that it provides higher reliability by flushing persistent memory stores to the most reliable persistence domain available to software rather than depending on automatic WPQ (write pending queue) flush on power failure (ADR).

The **pmem_deep_flush**() and **pmem_deep_drain**() functions provide partial varsions of **pmem_deep_persist**() function. **pmem_deep_persist**() can be thought of as this:

```
int pmem_deep_persist(const void *addr, size_t len)
{
    /* flush the processor caches */
    pmem_deep_flush(addr, len);

    /* wait for any pmem stores to drain from HW buffers */
    return pmem_deep_drain(addr, len);
}
```

Since this operation is usually much more expensive than **pmem_persist**(), it should be used rarely. Typically the application should use this function only to flush the most critical data, which are required to recover after the power failure.

The **pmem_has_auto_flush**() function checks if the machine supports automatic CPU cache flush on power failure or system crash. Function returns true only when each NVDIMM in the system is covered by this mechanism.

The **pmem_has_hw_drain**() function checks if the machine supports an explicit *hardware drain* instruction for persistent memory.

RETURN VALUE

The **pmem_persist**() function returns no value.

The **pmem_msync**() return value is the return value of **msync**(), which can return −1 and set *errno* to indicate an error.

The **pmem_flush**(), **pmem_drain**() and **pmem_deep_flush**() functions return no value.

The **pmem_deep_persist**() and **pmem_deep_drain**() return 0 on success. Otherwise it returns −1 and sets *errno* appropriately. If *len* is equal zero **pmem_deep_persist**() and **pmem_deep_drain**() return 0 but no flushing take place.

The **pmem_has_auto_flush**() function returns 1 if given platform supports processor cache flushing on a power loss event. Otherwise it returns 0. On error it returns −1 and sets *errno* appropriately.

The **pmem_has_hw_drain**() function returns true if the machine supports an explicit *hardware drain* instruction for persistent memory. On Intel processors with persistent memory, stores to persistent memory are considered persistent once they are flushed from the CPU caches, so this function always returns false. Despite that, programs using **pmem_flush**() to flush ranges of memory should still follow up by calling **pmem_drain**() once to ensure the flushes are complete. As mentioned above, **pmem_persist**() handles calling both **pmem_flush**() and **pmem_drain**().

SEE ALSO

msync(2), **pmem_is_pmem**(3), **libpmem**(7) and **<http://pmem.io>**