NAME

pmem_memmove(), **pmem_memcpy**(), **pmem_memset**(), **pmem_memmove_persist**(), **pmem_memcpy_persist**(), **pmem_memset_persist**(), **pmem_memmove_nodrain**(), **pmem_memcpy_nodrain**(), **pmem_memset_nodrain**() − functions that provide optimized copying to persistent memory

SYNOPSIS

```
#include <libpmem.h>

void *pmem_memmove(void *pmemdest, const void *src, size_t len, unsigned flags
void *pmem_memcpy(void *pmemdest, const void *src, size_t len, unsigned flags)
void *pmem_memset(void *pmemdest, int c, size_t len, unsigned flags);
void *pmem_memmove_persist(void *pmemdest, const void *src, size_t len);
void *pmem_memcpy_persist(void *pmemdest, const void *src, size_t len);
void *pmem_memset_persist(void *pmemdest, int c, size_t len);
void *pmem_memmove_nodrain(void *pmemdest, const void *src, size_t len);
void *pmem_memcpy_nodrain(void *pmemdest, const void *src, size_t len);
void *pmem_memset_nodrain(void *pmemdest, int c, size_t len);
```

DESCRIPTION

pmem_memmove(), **pmem_memcpy**() and **pmem_memset**() functions provide the same memory copying as their namesakes **memmove**(3), **memcpy**(3) and **memset**(3), and ensure that the result has been flushed to persistence before returning (unless **PMEM_F_MEM_NOFLUSH** flag was used).

For example, the following code is functionally equivalent to **pmem_memmove**() (with flags equal to 0):

```
memmove(dest, src, len);
pmem_persist(dest, len);
```

Calling **pmem_memmove**() may out−perform the above code, because **libpmem**(7) implementation may take advantage of the fact that *pmemdest* is persistent memory and use instructions such as *non−temporal* stores to avoid the need to flush processor caches.

> WARNING: Using these functions where **pmem_is_pmem**(3) returns false may not do anything useful. Use libc functions in that case.

Unlike libc implementation, **libpmem** functions guarantee that if destination buffer address and length are 8 byte aligned then all stores will be performed using at least 8 byte store instructions. This means that a series of 8 byte stores followed by **pmem_persist**(3) can be safely replaced by a single call to one of the above functions.

The *flags* argument of all of the above functions has the same meaning. It can be 0 or a bitwise OR of one or more of the following flags:

- **PMEM_F_MEM_NODRAIN** − modifies the behavior to skip the final **pmem_drain**() step. This allows applications to optimize cases where several ranges are being copied to persistent memory, followed by a single call to **pmem_drain**(). The following example illustrates how this flag might be used to avoid multiple calls to **pmem_drain**() when copying several ranges of memory to pmem:

```
/* ... write several ranges to pmem ... */
pmem_memcpy(pmemdest1, src1, len1, PMEM_F_MEM_NODRAIN);
pmem_memcpy(pmemdest2, src2, len2, PMEM_F_MEM_NODRAIN);

/* ... */

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

- **PMEM_F_MEM_NOFLUSH** − Don't flush anything. This implies **PMEM_F_MEM_NODRAIN**. Using this flag only makes sense when it's followed by any function that flushes data.

The remaining flags say *how* the operation should be done, and are merely hints.

- **PMEM_F_MEM_NONTEMPORAL** – Use non−temporal instructions. This flag is mutually exclusive with **PMEM_F_MEM_TEMPORAL**. On x86_64 this flag is mutually exclusive with **PMEM_F_MEM_NOFLUSH**.
- **PMEM_F_MEM_TEMPORAL** – Use temporal instructions. This flag is mutually exclusive with **PMEM_F_MEM_NONTEMPORAL**.
- **PMEM_F_MEM_WC** – Use write combining mode. This flag is mutually exclusive with **PMEM_F_MEM_WB**. On x86_64 this is an alias for **PMEM_F_MEM_NONTEMPORAL**. On x86_64 this flag is mutually exclusive with **PMEM_F_MEM_NOFLUSH**.
- **PMEM_F_MEM_WB** – Use write back mode. This flag is mutually exclusive with **PMEM_F_MEM_WC**. On x86_64 this is an alias for **PMEM_F_MEM_TEMPORAL**.

Using an invalid combination of flags has undefined behavior.

Without any of the above flags **libpmem** will try to guess the best strategy based on size. See **PMEM_MOVNT_THRESHOLD** description in **libpmem**(7) for details.

pmem_memmove_persist() is an alias for **pmem_memmove**() with flags equal to 0.

pmem_memcpy_persist() is an alias for **pmem_memcpy**() with flags equal to 0.

pmem_memset_persist() is an alias for **pmem_memset**() with flags equal to 0.

pmem_memmove_nodrain() is an alias for **pmem_memmove**() with flags equal to **PMEM_F_MEM_NODRAIN**.

pmem_memcpy_nodrain() is an alias for **pmem_memcpy**() with flags equal to **PMEM_F_MEM_NODRAIN**.

pmem_memset_nodrain() is an alias for **pmem_memset**() with flags equal to **PMEM_F_MEM_NODRAIN**.

RETURN VALUE

All of the above functions return address of the destination buffer.

CAVEATS

After calling any of the functions with **PMEM_F_MEM_NODRAIN** flag you should not expect memory to be visible to other threads before calling **pmem_drain**(3) or any of the *_persist* functions. This is because on x86_64 those functions may use non−temporal store instructions, which are weakly ordered. See "Intel 64 and IA−32 Architectures Software Developer's Manual", Volume 1, "Caching of Temporal vs. Non−Temporal Data" section for details.

SEE ALSO

memcpy(3), **memmove**(3), **memset**(3), **libpmem**(7) and **<http://pmem.io>**