Library Routines

Initialization Routines

```c
void start_pes(int npes);
```
Initializes the OpenSHMEM library. This routine must be called before any library other routine is called.

Query Routines

```c
int _my_pe(void);
```
Returns the virtual PE number of the calling PE.

```c
int _num_pes(void);
```
Returns the virtual PE number of the calling PE.

Data Transfer Routines

```c
void shmem_[funcname]_g([type] *addr, int pe);
```
Retrieve data of basic types from a remote PE.
[funcname] can be anything in { short, int, float, double, long }
[type] can be anything in { short, int, float, double, long }

```c
void shmem_[funcname]_get([type] *dest, [type] *src, size_t len, int pe);
```
Retrieve contiguous data from a remote PE.
[funcname] can be anything in { short, int, float, double, long, longlong, longdouble }
[type] can be anything in { short, int, float, double, long, long long, long double }

```c
void shmem_get_[funcname](void *dest, void *src, size_t len, int pe);
```
Retrieve contiguous data from a remote PE.
[funcname] can be anything in { 32, 64, 128, mem }

```c
void shmem_[funcname]_iget([type] *dest, const [type] *src, ptrdiff_t tst, ptrdiff_t sst, size_t len, int pe);
```
Retrieve strided (target, source stride can be different) data from a remote PE.
[funcname] can be anything in { short, int, float, double, long, longlong, longdouble }
[type] can be anything in { short, int, float, double, long, long long, long double }
Data Transfer Routines (Continued)

**void shmem_[funcname]_p([type] *addr, int pe);**
Write data of basic types to a remote PE.
[funcname] can be any of { short, int, float, double, long }
[type] can be any of { short, int, float, double, long }

**void shmem_[funcname]_put([type] *dest, [type] *src, size_t len, int pe);**
Write contiguous data to a remote PE.
[funcname] can be any of { short, int, float, double, long, longlong, longdouble }
[type] can be any of { short, int, float, double, long, long long, long double }

**void shmem_put[funcname](void *dest, void *src, size_t len, int pe);**
Write contiguous data to a remote PE.
[funcname] can be any of { 32, 64, 128, mem }

**void shmem_[funcname]_iput([type] *dest, const [type] *src, ptrdiff_t tst, ptrdiff_t sst, size_t len, int pe);**
Write strided (target, source stride can be different) data to a remote PE.
[funcname] can be any of { short, int, float, double, long, longlong, longdouble }
[type] can be any of { short, int, float, double, long, long long, long double }

Synchronization Routines

**void shmem_barrier_all(void);**
Suspend execution on the calling PE, until all other PEs reach this point of execution path.

**void shmem_barrier(int PE_start, int logPE_stride, int PE_size, long *pSync);**
Suspend execution on the calling PE, until a subset of PEs, defined by PE_start, logPE_stride and PE_size, reaches this point of execution path.

**void shmem_fence(void);**
Ensure ordering or remote put operations to a particular PE.

**void shmem_quiet(void);**
Ensure ordering or remote put operations to multiple Pes.

Symmetric Heap Routines

**void *shmalloc(size_t size);**
Allocates a memory block in the symmetric heap.

**void *shrealloc(void *ptr, size_t size);**
Adjust the size of a symmetric memory block.
Symmetric Heap Routines (Continued)

void shfree(void *ptr);
Deallocates a symmetric memory block.

void *shmemalign(size_t alignment, size_t size);
Returns a symmetric memory block aligned with to the size specified by alignment.

Remote Pointer Routines

void *shmem_ptr(void *target, int pe);
Returns a pointer to a data object of a remote PE.

Collect Routines

void shmem_fcollect[bits](void *target, const void *source,
    size_t nlong, int PE_start, int logPE_stride, int PE_size,
    long *pSync);
Concatenate remote data objects and stores the result in a local data object. nlong must be
the same on all PEs.
[bits] can be any of { 32, 64 }

void shmem_collect[bits](void *target, const void *source,
    size_t nlong, int PE_start, int logPE_stride, int PE_size,
    long *pSync);
Concatenate remote data objects and stores the result in a local data object. nlong can vary
from PE to PE.
[bits] can be any of { 32, 64 }

Broadcast Routines

void shmem_broadcast[bits](void *target, const void *source,
    size_t nlong, int PE_root, int PE_start, int logPE_stride,
    int PE_size, long *pSync);
Write data to a symmetric data object on all PEs of the active set.
[bits] can be any of { 32, 64 }

Reduction Routines

void shmem_[funcname]_[opname]_to_all([type] *target, [type] *source, int nreduce, int PE_start, int logPE_stride,
    int PE_size, int *pWrk, long *pSync);
Perform a reduction operation on symmetric data objects of all PEs in the active set.
[funcname] can be any of { short, int, float, double, long, longlong, longdouble }
[opname] can be any of { and, or, xor, sum, prod, max, min }
[type] can be any of { short, int, float, double, long, long long, long double }
Environment Variables

SGI Specific Environment Variables

**SMA_VERSION**
Print library version at library startup.

**SMA_INFO**
Print helpful text about all these environment variables.

**SMA_SYMMETRIC_SIZE**
Number of bytes to allocate for the symmetric heap.

**SMA_DEBUG**
Enable debugging messages.

Reference Implementation Specific Environment Variables

**SHMEM_LOG_LEVELS**
A comma, space, semi-colon separated list of logging/trace facilities to enable debugging messages. The facilities currently supported include the following case-sensitive names:

- FATAL
- DEBUG
- INFO
- NOTICE
- AUTH
- INIT
- MEMORY
- CACHE
- BARRIER
- BROADCAST
- COLLECT
- REDUCE
- SYMBOLS
- LOCK
- SERVICE
- FENCE
- QUIET

Please refer to the OpenSHMEM Reference Implementation design document for more information about the facilities mentioned above.

**SHMEM_LOG_FILE**
A filename to which to write log messages.

**SHMEM_SYMMETRIC_HEAP_SIZE**
The number of bytes to allocate for the symmetric heap area. Can scale units with “K”, “M” etc. modifiers. The default is 1M.

**SHMEM_BARRIER_ALGORITHM**
The version of the barrier to use. The default is “naive”. Designed to allow people to plug other variants in easily and test.

**SHMEM_BARRIER_ALGORITHM_ALL**
As for **SHMEM_BARRIER_ALGORITHM**, but separating these two allows us to optimize if e.g. hardware has special support for global barriers.

**SHMEM_PE_ACCESSIBLE_TIMEOUT**
The number of seconds to wait for PEs to reply to accessibility checks. The default is 1.0 (i.e. may be fractional).