SHMEM Run-Time Error Detection Test Suite for C

Iowa State University’s High Performance Computing Group
February 24, 2010

James Coyle, Senior Scientist
James Hoekstra, Senior Scientist
Glenn Luecke, Director
Marina Kraeva, Senior Scientist
Zachary Voller, Graduate Student

TABLE OF CONTENTS
Section A. Deadlocks caused by incorrect usage of SHMEM routines ........................................... 2
Section B. SHMEM Environment Variables.................................................................................................. 4
Section C. Incorrect order of SHMEM routines calls .................................................................................... 5
Section D. Uninitialized variables in SHMEM routines calls ................................................................. 7
Section E. Argument Errors in SHMEM routines calls ........................................................................... 7
Section F. Non portable usage of SHMEM ................................................................................................... 13
Section G. SHMEM Errors Causing Race Conditions .............................................................................. 13

Remarks
These tests will be designed to evaluate the ability of tools to detect SHMEM errors at run-time. Each test program contains one and only one SHMEM error.

Methodology
Tests for each of the sections below will be developed by carefully going through each section in the SHMEM API and writing appropriate tests and error message files. Tests and error message files will be carefully reviewed by a different person. Tests will be developed on a Cray system and then run on other machines.

Unless specified otherwise, tests will be designed with the number of threads as a parameter to allow for easy changing of the number of threads.

To prevent dead code elimination by the compiler, the appropriate variables will be printed.
Section A. Deadlocks caused by incorrect usage of SHMEM routines

A.1. Not every process in the active group calls a synchronization routine
   (a) shmem_barrier_all
   (b) shmem_barrier
   (c) barrier

A.2. Deadlock at the call to a wait routine

A.2.1. call a shmem_*_wait routine on one process (set ivar to value, never update ivar by other processes)
   (a) shmem_int_wait
   (b) shmem_long_wait
   (c) shmem_longlong_wait
   (d) shmem_short_wait
   (e) shmem_wait

A.2.2. call a shmem_*_wait routine on one process (set ivar to value, update ivar to the same value by another process, don’t do other updates)
   (a) shmem_int_wait
   (b) shmem_long_wait
   (c) shmem_longlong_wait
   (d) shmem_short_wait
   (e) shmem_wait

A.2.3. call a shmem_*_wait_until routine with cond=SHMEM_CMP_NE on one process (set ivar to value, never update ivar by other processes)
   (a) shmem_int_wait_until
   (b) shmem_long_wait_until
   (c) shmem_longlong_wait_until
   (d) shmem_short_wait_until
   (e) shmem_wait_until

A.2.4. call a shmem_*_wait_until routine with cond=SHMEM_CMP_NE on one process (set ivar to value, update ivar to the same value by another process, don’t do other updates)
   (a) shmem_int_wait_until
   (b) shmem_long_wait_until
   (c) shmem_longlong_wait_until
   (d) shmem_short_wait_until
   (e) shmem_wait_until
A.2.5. call a shmem_*_wait_until routine with cond=SHMEM_CMP_EQ on one process (set ivar to value-1, never update ivar by other processes)
   (a) shmem_int_wait_until
   (b) shmem_long_wait_until
   (c) shmem_longlong_wait_until
   (d) shmem_short_wait_until
   (e) shmem_wait_until

A.2.6. call a shmem_*_wait_until routine with cond=SHMEM_CMP_EQ on one process (set ivar to value-1, update ivar to value-2 by another process, don’t do other updates)
   (a) shmem_int_wait_until
   (b) shmem_long_wait_until
   (c) shmem_longlong_wait_until
   (d) shmem_short_wait_until
   (e) shmem_wait_until

A.2.7. call a shmem_*_wait_until routine with cond=SHMEM_CMP_GT on one process (set ivar to value, never update ivar by other processes)
   (a) shmem_int_wait_until

A.2.8. call a shmem_*_wait_until routine with cond=SHMEM_CMP_GT on one process (set ivar to value, keep updating ivar to lesser values by other processes)
   (a) shmem_long_wait_until

A.2.9. call a shmem_*_wait_until routine with cond=SHMEM_CMP_GE on one process (set ivar to value-1, never update ivar by other processes)
   (a) shmem_longlong_wait_until

A.2.10.call a shmem_*_wait_until routine with cond=SHMEM_CMP_GE on one process (set ivar to value-1, keep updating ivar to lesser values by other processes)
   (a) shmem_short_wait_until

A.2.11.call a shmem_*_wait_until routine with cond=SHMEM_CMP_LT on one process (set ivar to value, never update ivar by other processes)
   (a) shmem_wait_until

A.2.12.call a shmem_*_wait_until routine with cond=SHMEM_CMP_LT on one process (set ivar to value, keep updating ivar to larger values by other processes)
   (a) shmem_int_wait_until

A.2.13.call a shmem_*_wait_until routine with cond=SHMEM_CMP_LE on one process (set ivar to value+1, never update ivar by other processes)
   (a) shmem_long_wait_until
A.2.14. call a shmem_*_wait_until routine with cond=SHMEM_CMP_LE on one process (set ivar to value+1, keep updating ivar to larger values by other processes)
   (a) shmem_longlong_wait_until

A.3. Not every process in the active group calls a collective routine
   (a) shmem_min
   (b) shmem_or
   (c) shmem_prod
   (d) shmem_sum
   (e) shmem_xor

Section B. SHMEM Environment Variables

The following environment variables affect SHMEM behavior.

**XT_LINUX_SHMEM_STACK_SIZE**
   CNL only. Controls the size (in bytes) of the stack. A process cannot request memory for the stack in excess of the total memory on a node divided by the number of CPUs on that node.
   Default: 16 MB

**XT_LINUX_SHMEM_HEAP_SIZE**
   CNL only. Controls the size (in bytes) of the private heap. A process cannot request memory for the heap in excess of the total memory on a node divided by the number of CPUs on that node.
   Default: 100 MB

**XT_SYMMETRIC_HEAP_SIZE**
   Controls the size (in bytes) of the symmetric heap.
   Defaults: 0 (Catamount), 2MB (CNL)

**SHMEM_SWAP_BACKOFF**
   CNL only. Set the backoff algorithm multiplier value.
   Default: 100

**PMI_EXIT_QUIET**
   Enable or disable printing of debugging information when a rank in the job terminates abnormally.
   Default: enabled

B.1. Request too much memory by setting
   (a) XT_LINUX_SHMEM_STACK_SIZE be larger than the total memory on a node divided by the number of CPUs on that node.
(b) XT_LINUX_SHMEM_HEAP_SIZE be larger than the total memory on a node divided by the number of CPUs on that node.
(c) XT_LINUX_SHMEM_STACK_SIZE and XT_LINUX_SHMEM_HEAP_SIZE each be larger than the half of the total memory on a node divided by the number of CPUs on that node but less than the total memory on a node divided by the number of CPUs on that node.

Section C. Incorrect order of SHMEM routines calls

In all the tests in this section the SHMEM calls will be placed into different translation units in separate files to prevent error detection at compile time.

C.1. Call start_pes or shmem_init twice
   (a) two subsequent calls to start_pes
   (b) two subsequent calls to shmem_init
   (c) subsequent calls to start_pes, then to shmem_init
   (d) subsequent calls to shmem_init, then to start_pes

C.2. Call a SHMEM routine before the call to start_pes or shmem_init
A SHMEM application must call start_pes or shmem_init as the very first SHMEM routine called within the application to guarantee that lower-level resources have been set up correctly.

C.2.1. Symmetric Heap Management
   (a) shmalloc
   (b) shrealloc
   (c) shfree

C.2.2. PE Queries
   (a) shmem_my_pe
   (b) shmem_n_pes

C.2.3. Block Data Get
   (a) shmem_get
   (b) shmem_get16
   (c) shmem_get32
   (d) shmem_get64
   (e) shmem_get128
   (f) shmem_short_get
   (g) shmem_int_get
   (h) shmem_long_get
   (i) shmem_longlong_get
   (j) shmem_float_get
   (k) shmem_double_get
C.2.4. Block Data Put  
(a) shmem_put  
(b) shmem_put16  
(c) shmem_put32  
(d) shmem_put64  
(e) shmem_put128  
(f) shmem_short_put  
(g) shmem_int_put  
(h) shmem_long_put  
(i) shmem_longlong_put  
(j) shmem_float_put  
(k) shmem_double_put  

C.3. Call a SHMEM routine after the call to shmem_finalize  
A SHMEM application must call shmem_finalize as the very last SHMEM routine called within the application to guarantee correct clean-up of previously allocated network protocol resources.  

C.3.1. Symmetric Heap Management  
(a) shmalloc  
(b) shrealloc  
(c) shfree  

C.3.2. PE Queries  
(a) shmem_my_pe  
(b) shmem_n_pes  

C.3.3. Block Data Get  
(a) shmem_get  
(b) shmem_get16  
(c) shmem_get32  
(d) shmem_get64  
(e) shmem_get128  
(f) shmem_short_get  
(g) shmem_int_get  
(h) shmem_long_get  
(i) shmem_longlong_get  
(j) shmem_float_get  
(k) shmem_double_get  

C.3.4. Block Data Put  
(a) shmem_put  
(b) shmem_put16  
(c) shmem_put32  
(d) shmem_put64  
(e) shmem_put128  
(f) shmem_short_put
Section D. Uninitialized variables in SHMEM routines calls

Section E. Argument Errors in SHMEM routines calls

E.1. Pass non-symmetric data objects as arguments that are required to be remotely accessible.

E.1.1. Pass stack variables

E.1.1.1. as the source argument in Get routines

(a) shmem_get
(b) shmem_get16
(c) shmem_get32
(d) shmem_get64
(e) shmem_get128
(f) shmem_short_get
(g) shmem_int_get
(h) shmem_long_get
(i) shmem_longlong_get
(j) shmem_float_get
(k) shmem_double_get
(l) shmem_getmem
(m)shmem_iget
(n) shmem_iget16
(o) shmem_iget32
(p) shmem_iget64
(q) shmem_iget128
(r) shmem_short_iget
(s) shmem_int_iget
(t) shmem_long_iget
(u) shmem_longlong_iget
(v) shmem_float_iget
(w)shmem_double_iget
(x) shmem_char_g
(y) shmem_short_g
(z) shmem_int_g  
(aa) shmem_long_g  
(bb) shmem_longlong_g  
(cc) shmem_float_g  
(dd) shmem_double_g  
(ee) shmem_longdouble_g  

E.1.1.2. as the target argument in Put routines  
(a) shmem_put  
(b) shmem_put16  
(c) shmem_put32  
(d) shmem_put64  
(e) shmem_put128  
(f) shmem_short_put  
(g) shmem_int_put  
(h) shmem_long_put  
(i) shmem_longlong_put  
(j) shmem_float_put  
(k) shmem_double_put  
(l) shmem_putmem  
(m) shmem_iput  
(n) shmem_iput16  
(o) shmem_iput32  
(p) shmem_iput64  
(q) shmem_iput128  
(r) shmem_short_iput  
(s) shmem_int_iput  
(t) shmem_long_iput  
(u) shmem_longlong_iput  
(v) shmem_float_iput  
(w) shmem_double_iput  
(x) shmem_char_p  
(y) shmem_short_p  
(z) shmem_int_p  
(aa) shmem_long_p  
(bb) shmem_longlong_p  
(cc) shmem_float_p  
(dd) shmem_double_p  
(ee) shmem_longdouble_p  

E.1.1.3. as the source argument in collective routines  
(a) shmem_double_min_to_all  
(b) shmem_float_min_to_all  
(c) shmem_int_min_to_all  
(d) shmem_long_min_to_all  
(e) shmem_longdouble_min_to_all  
(f) shmem_longlong_min_to_all
(g) shmem_short_min_to_all
(h) shmem_double_max_to_all
(i) shmem_float_max_to_all
(j) shmem_int_max_to_all
(k) shmem_long_max_to_all
(l) shmem_longdouble_max_to_all
(m) shmem_longlong_max_to_all
(n) shmem_short_max_to_all
(o) shmem_int_or_to_all
(p) shmem_long_or_to_all
(q) shmem_longlong_or_to_all
(r) shmem_short_or_to_all
(s) shmem_int_xor_to_all
(t) shmem_long_xor_to_all
(u) shmem_longlong_xor_to_all
(v) shmem_short_xor_to_all
(w) shmem_int_and_to_all
(x) shmem_long_and_to_all
(y) shmem_longlong_and_to_all
(z) shmem_short_and_to_all
(aa) shmem_double_sum_to_all
(bb) shmem_float_sum_to_all
(cc) shmem_int_sum_to_all
(dd) shmem_long_sum_to_all
(ee) shmem_longdouble_sum_to_all
(ff) shmem_longlong_sum_to_all
(gg) shmem_short_sum_to_all
(hh) shmem_complexd_sum_to_all
(ii) shmem_complexf_sum_to_all
(jj) shmem_double_prod_to_all
(kk) shmem_float_prod_to_all
(ll) shmem_int_prod_to_all
(mm) shmem_long_prod_to_all
(nn) shmem_longdouble_prod_to_all
(oo) shmem_longlong_prod_to_all
(pp) shmem_short_prod_to_all
(qq) shmem_complexd_prod_to_all
(rr) shmem_complexf_prod_to_all

E.1.1.4. as the target argument in collective routines

(a) shmem_double_min_to_all
(b) shmem_float_min_to_all
(c) shmem_int_min_to_all
(d) shmem_long_min_to_all
(e) shmem_longdouble_min_to_all
(f) shmem_longlong_min_to_all
(g) shmem_short_min_to_all
(h) shmem_double_max_to_all
(i) shmem_float_max_to_all
(j) shmem_int_max_to_all
(k) shmem_long_max_to_all
(l) shmem_longdouble_max_to_all
(m) shmem_longlong_max_to_all
(n) shmem_short_max_to_all
(o) shmem_int_or_to_all
(p) shmem_long_or_to_all
(q) shmem_longlong_or_to_all
(r) shmem_short_or_to_all
(s) shmem_int_xor_to_all
(t) shmem_long_xor_to_all
(u) shmem_longlong_xor_to_all
(v) shmem_short_xor_to_all
(w) shmem_int_and_to_all
(x) shmem_long_and_to_all
(y) shmem_longlong_and_to_all
(z) shmem_short_and_to_all
(aa) shmem_double_sum_to_all
(bb) shmem_float_sum_to_all
(cc) shmem_int_sum_to_all
(dd) shmem_long_sum_to_all
(ee) shmem_longdouble_sum_to_all
(ff) shmem_longlong_sum_to_all
(gg) shmem_short_sum_to_all
(hh) shmem_complexd_sum_to_all
(ii) shmem_complexf_sum_to_all
(jj) shmem_double_prod_to_all
(kk) shmem_float_prod_to_all
(ll) shmem_int_prod_to_all
(mm) shmem_long_prod_to_all
(nn) shmem_longdouble_prod_to_all
(oo) shmem_longlong_prod_to_all
(pp) shmem_short_prod_to_all
(qq) shmem_complexd_prod_to_all
(rr) shmem_complexf_prod_to_all

E.1.1.5. as the pWrk argument in collective routines
(a) shmem_double_min_to_all
(b) shmem_float_min_to_all
(c) shmem_int_min_to_all
(d) shmem_long_min_to_all
(e) shmem_longdouble_min_to_all
(f) shmem_longlong_min_to_all
(g) shmem_short_min_to_all
(h) shmem_double_max_to_all
E.1.2. Pass dynamically allocated arrays (by a call to malloc)

E.1.2.1. as the source argument in Get routines
   (a) shmem_get
   (b) shmem_get16
   (c) shmem_get32
   (d) shmem_get64
   (e) shmem_get128
   (f) shmem_short_get
   (g) shmem_int_get
(h) shmem_long_get
(i) shmem_longlong_get
(j) shmem_float_get
(k) shmem_double_get
(l) shmem_getmem
(m) shmem_iget
(n) shmem_iget16
(o) shmem_iget32
(p) shmem_iget64
(q) shmem_iget128
(r) shmem_short_iget
(s) shmem_int_iget
(t) shmem_long_iget
(u) shmem_longlong_iget
(v) shmem_float_iget
(w) shmem_double_iget

E.1.2.2. as the target argument in Put routines
   (a) shmem_put
   (b) shmem_put16
   (c) shmem_put32
   (d) shmem_put64
   (e) shmem_put128
   (f) shmem_short_put
   (g) shmem_int_put
   (h) shmem_long_put
   (i) shmem_longlong_put
   (j) shmem_float_put
   (k) shmem_double_put
   (l) shmem_putmem
   (m) shmem_iput
   (n) shmem_iput16
   (o) shmem_iput32
   (p) shmem_iput64
   (q) shmem_iput128
   (r) shmem_short_iput
   (s) shmem_int_iput
   (t) shmem_long_iput
   (u) shmem_longlong_iput
   (v) shmem_float_iput
   (w) shmem_double_iput

E.2. Use source and target arguments of different data types.
The source argument must have the same data type as target.
E.3. Use pWrk and target arguments of different data types.
The pWrk argument must have the same data type as target.

E.4. Pass different values for the arguments that are required to be single valued.

E.4.1. In shmem_barrier on different processes in the active set use different values of
(a) PE_start
(b) logPE_stride
(c) PE_size

E.5. Use overlapping (but not the same) arrays as source and target arguments.
The source and target arrays may be the same array, but they may not be overlapping arrays.

Section F. Non portable usage of SHMEM

1. The start_pes routine only validates its argument.
   For UNICOS/mp, the number of PEs is specified by the -n option on the aprun/mpirun command.
The -n option must match the compile time -X option, if the -X option is specified.
   For CLE, the number of PEs is specified by the -size, -sz, or -np option on the yod command.

2. The start_pes routine accepts the following argument:
   npes Identifies the total number of PEs desired. If npes is 0, the number of PEs is selected
   by the NPES environment variable.

3. Programs that use both MPI and SHMEM should call MPI_Init followed by start_pes or
   shmemb_init. At the end of the program, shmemb_finalize should be called followed by
   MPI_Finalize.

4. If a PE not in the active set calls a SHMEM collective routine, undefined behavior results.

Section G. SHMEM Errors Causing Race Conditions