1 Hands-on; Shared Memory III; MPI with Pthread, OpenMP

1. **Computes the dot product of two vectors**;

   - Serial - Serial program.
   - Pthreads only - A shared memory programming model using Pthreads.
   - MPI only - A distributed memory programming model with MPI.
   - MPI with pthreads - A hybrid model that utilizes both MPI and Pthreads to execute on systems that are comprised of clusters of SMP's.
   - Makefile - Use as

     ```
     make -f makecode5154
     ```

   In general, there may be problems if multiple threads make MPI calls. The program may fail or behave unexpectedly. If MPI calls must be made from within a thread, they should be made only by one thread.

2. **Another example (program)** for MPI with Pthread.

3. **OpenMP**. Two example programs are given; hello, workshare. Follow the steps below before executing OpenMP codes;

   ```
   export PGI=/usr/local/pgi
   export PATH=$PGI/linux86/6.2/bin:$PATH
   export MANPATH=$MANPATH:$PGI/linux86/6.2/man
   export LD_LIBRARY_PATH=/usr/local/pgi/linux86/6.2/liblf:
   /usr/local/pgi/linux86/6.2/lib:$LD_LIBRARY_PATH
   ```

4. **Hello world**

   - In this simple example, the master thread forks a parallel region.
   - All threads in the team obtain their unique thread number and print it.
   - The master thread only prints the total number of threads. Two OpenMP library routines are used to obtain the number of threads and each thread's number.

     ```
     export OMP_NUM_THREADS=4
     pgcc -o omp_hello code25.c -mp
     ```

5. **Loop work-sharing**

    - The iterations of a loop are scheduled dynamically across the team of threads.
    - A thread will perform CHUNK iterations at a time before being scheduled for the next CHUNK of work.

    ```
    pgcc -o omp_workshare code26.c -mp
    ```

1.1 HOMEWORK - Due to December 27, 2010

1. **Finding the minimum of a list of integers with threads**. Complete this template;

    - The list is partitioned equally among the threads.
    - The size of each thread's partition is stored in the variable (*partial_list_size*).
    - The pointer to the start of each thread's partial list is passed to it as the pointer (*list_ptr*).
    - The test-update operation for *minimum_value* is protected by the mutex-lock *minimum_value_lock*.
    - Threads execute *pthread_mutex_lock* to gain exclusive access to the variable *minimum_value*.
    - Once this access is gained, the value is updated as required, and the lock subsequently released.
    - Since at any point of time, only one thread can hold a lock, only one thread can test-update the variable.