Christopher Bero
CPE 412
HW 1

Problem 1

(A)

- hello_world_MPI outputs in a different order each time.
- hello_world_PTH outputs in a different order each time.
- hello_world_OMP outputs in a different order each time.

hello_world_MPI output

```
uahcls16@dmc:hw1> mpirun -np 8 ./hw_mpi
--------------------------------------------------------------------------
WARNING: It appears that your OpenFabrics subsystem is configured to only
allow registering part of your physical memory.  This can cause MPI jobs to
run with erratic performance, hang, and/or crash.

This may be caused by your OpenFabrics vendor limiting the amount of
physical memory that can be registered.  You should investigate the
relevant Linux kernel module parameters that control how much physical
memory can be registered, and increase them to allow registering all
physical memory on your machine.

See this Open MPI FAQ item for more information on these Linux kernel module
parameters:

    http://www.open-mpi.org/faq/?category=openfabrics#ib-locked-pages

  Local host:              dmc
  Registerable memory:     16384 MiB
  Total memory:            24567 MiB

Your MPI job will continue, but may be behave poorly and/or hang.
--------------------------------------------------------------------------
Hello World from MPI Process #4
Hello World from MPI Process #6
Hello World from MPI Process #7
Hello World from MPI Process #3
Hello World from MPI Process #0
Number of MPI Processes = 8
Hello World from MPI Process #2
Hello World from MPI Process #5
Hello World from MPI Process #1
[dmc:24221] 7 more processes have sent help message help-mpi-btl-openib.txt / reg
mem limit low
[dmc:24221] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help /
error messages
```

hello_world_MPI2 output

```
uahcls16@dmc:hw1> mpirun -np 8 ./hw_mpi2
--------------------------------------------------------------------------
WARNING: It appears that your OpenFabrics subsystem is configured to only
allow registering part of your physical memory.  This can cause MPI jobs to
run with erratic performance, hang, and/or crash.

This may be caused by your OpenFabrics vendor limiting the amount of
physical memory that can be registered.  You should investigate the
relevant Linux kernel module parameters that control how much physical
memory can be registered, and increase them to allow registering all
physical memory on your machine.

See this Open MPI FAQ item for more information on these Linux kernel module
parameters:
    http://www.open-mpi.org/faq/?category=openfabrics#ib-locked-pages

  Local host:              dmc
  Registerable memory:     16384 MiB
  Total memory:            24567 MiB

Your MPI job will continue, but may be behave poorly and/or hang.
--------------------------------------------------------------------------
Hello World from MPI Process #0
Number of MPI Processes = 8
Hello World from MPI Process #1
Hello World from MPI Process #5
Hello World from MPI Process #3
Hello World from MPI Process #2
Hello World from MPI Process #6
Hello World from MPI Process #7
Hello World from MPI Process #4
[dmc:24728] 7 more processes have sent help message help-mpi-btl-openib.txt / reg
mem limit low
[dmc:24728] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help /
error messages
```

hello_world_PTH output

```
uahcls16@dmc:hw1> ./hw_pth
Hello World, from PThread 0
Hello World, from PThread 1
Hello World, from PThread 7
Hello World, from PThread 2
Hello World, from PThread 5
Hello World, from PThread 4
Number of threads = 8
Hello World, from PThread 6
Hello World, from PThread 3
```

hello_world_OMP output

```
uahcls16@dmc:hw1> ./hw_omp
Hello World from thread = 1
Hello World from thread = 0
Hello World from thread = 7
Hello World from thread = 5
Hello World from thread = 2
Hello World from thread = 3
Hello World from thread = 6
Hello World from thread = 4
Number of Threads = 8
```

(B)
- When the mutex lock and unlock lines are commented out of the pthreads program, its output will clobber itself, and result in some half-lines and run-on lines being printed. I suspect this is because there is only one stdout, and when multiple threads attempt to write output, it simply clobbers and continues.

hello_world_PTH_mod output

```
uahcls16@dmc:hw1> ./hw_pth_mod
Hello World, from PThread Hello World, from PThread 0Hello World, from PThread 41

Hello World, from PThread Hello World, from PThread Number of threads = 8
Hello World, from PThread 7

5
3
```

```
Hello World, from PThread 2
Hello World, from PThread 6
```

(C)
- Similarly to the modified Pthread program, the OpenMP executable will output nothing on some lines, garbage on others, and print a few that match the original. Again I believe this is because the program's threads are attmpting to access the same stdout at the same time, and the buffer is not cleared to the screen before being overwritten.

hello_world_OMP_mod output

```
uahcls16@dmc:hw1> ./hw_omp_mod
Hello World from thread = Hello World from thread = Hello World from thread =
Hello World from thread = 763
Hello World from thread = Hello World from thread = 4
Hello World from thread = 0
Number of Threads = 8
Hello World from thread = 2
5

1
```

Problem 2

hw1p2 source

```
/*
Christopher Bero
CPE 412
Homework 1 - Problem 2
*/

using namespace std;

#include <iostream>
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>

#define NTHREADS 2 /* set number of threads to 8 */

pthread_mutex_t MUTEX; // globaly defined lock variable used to insure
                       // uninterrupted print operations from each thread

struct arg_struct {
        int proc;
        int thread;
};

// PThread - This is the worker thread code
void *hello(void * arg) {
        struct arg_struct *args_p = (struct arg_struct *)arg;

   // enter critical region
   pthread_mutex_lock(&MUTEX);

   cout << "Hello World, from MPI Process " << args_p->proc << " Thread " <<
args_p->thread << endl << flush;

   // exit critical region
   pthread_mutex_unlock(&MUTEX);

  return arg;
}

int main (int argc, char *argv[]) {
  int tid;                        // tid thread number
  pthread_t threads[NTHREADS];    // holds various thread info
  int errcode;                    // pthread error code
   MPI_Status status;
   int nmtsks, rank;

   MPI_Init(&argc,&argv); // Initalize MPI environment
   MPI_Comm_size(MPI_COMM_WORLD,&nmtsks); //get total number of processes
   MPI_Comm_rank(MPI_COMM_WORLD,&rank); // get process identity number

  //int ids[nmtsks][2];              // holds thread args

        arg_struct args[nmtsks][NTHREADS];
   // initialize mutex variable -- this variable is used to insure that
   // all couts are automic meaning that they are not interrupted
   pthread_mutex_init(&MUTEX,NULL);

   /* create the threads */
   for (tid=0; tid<NTHREADS; tid++) {
        args[rank][tid].proc=rank;
      args[rank][tid].thread=tid;
      errcode=pthread_create(
              &threads[tid],// thread information structure
              NULL,         // thread attributes -- NULL means assume defaults
              hello,        // function name that is to represent thread
              (void*)&args[rank][tid]);  // pthread created thread id for the
created thread
      // check for error during thread creation
      if (errcode) {
         cerr << "Pthread Creation Error: " << strerror(errcode) << endl << flush;
         exit(1);
      }
   }

   // wait for all threads to return before exiting main program (process)
   for (tid=0; tid<NTHREADS; tid++) {
      // wait for each thread to terminate
      errcode=pthread_join(
              threads[tid], //thread scheduler id information of selected thread
              NULL);        //thread return value -- not used in this case
      if (errcode) {
         cerr << "Pthread Join Error: " << strerror(errcode) << endl << flush;
         exit(1);
      }
   }

  /* Terminate MPI Program -- clear out all buffers */
  MPI_Finalize();

}
```

hw1p2 output

```
uahcls16@dmc:hw1> mpiexec -np 4 ./hybrid
--------------------------------------------------------------------------
WARNING: It appears that your OpenFabrics subsystem is configured to only
allow registering part of your physical memory.  This can cause MPI jobs to
run with erratic performance, hang, and/or crash.

This may be caused by your OpenFabrics vendor limiting the amount of
physical memory that can be registered.  You should investigate the
relevant Linux kernel module parameters that control how much physical
memory can be registered, and increase them to allow registering all
physical memory on your machine.

See this Open MPI FAQ item for more information on these Linux kernel module
parameters:

    http://www.open-mpi.org/faq/?category=openfabrics#ib-locked-pages

  Local host:              dmc
  Registerable memory:     16384 MiB
  Total memory:            24567 MiB

Your MPI job will continue, but may be behave poorly and/or hang.
--------------------------------------------------------------------------
Hello World, from MPI Process 2 Thread 0
Hello World, from MPI Process 2 Thread 1
Hello World, from MPI Process 0 Thread 0
Hello World, from MPI Process 0 Thread 1
Hello World, from MPI Process 1 Thread 0
Hello World, from MPI Process 3 Thread 1
Hello World, from MPI Process 3 Thread 0
Hello World, from MPI Process 1 Thread 1
[dmc:04740] 3 more processes have sent help message help-mpi-btl-openib.txt / reg
mem limit low
[dmc:04740] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help /
error messages
```