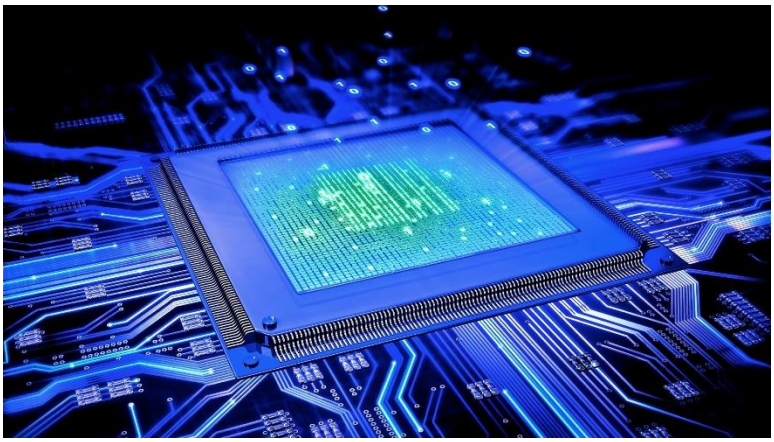

Homework #1

CPE 512

KYLE RAY

August 31, 2017

Contents

Example Program Output

Below are the results of running each example program in the DMC environment.

MPI

```
uahcls01@dmcvlogin1:Hw1> mpirun hello_world_MPI
Hello World from MPI Process #0
Number of MPI Processes = 8
Hello World from MPI Process #1
Hello World from MPI Process #6
Hello World from MPI Process #7
Hello World from MPI Process #2
Hello World from MPI Process #3
Hello World from MPI Process #5
Hello World from MPI Process #4
```

MPI2

```
uahcls01@dmcvlogin1:Hw1> mpirun hello_world_MPI2
Hello World from MPI Process #1
Hello World from MPI Process #3
Hello World from MPI Process #5
Hello World from MPI Process #6
Hello World from MPI Process #0
Number of MPI Processes = 8
Hello World from MPI Process #7
Hello World from MPI Process #4
Hello World from MPI Process #2
```

PTHREAD

```
uahcls01@dmcvlogin1:Hw1> ./hello_world_PTH
Hello World, from PThread 0
Number of threads = 8
Hello World, from PThread 2
Hello World, from PThread 3
Hello World, from PThread 4
Hello World, from PThread 5
Hello World, from PThread 6
Hello World, from PThread 1
Hello World, from PThread 7
```

OPENMP

```
uahcls01@dmcvlogin1:Hw1> ./hello_world_OMP
Hello World from thread = 0
Hello World from thread = 5
Hello World from thread = 3
Number of Threads = 8
Hello World from thread = 7
Hello World from thread = 4
Hello World from thread = 1
Hello World from thread = 2
Hello World from thread = 6
```

Questions

1.) Do the outputs always appear the same each time the program is executed?
   a. The outputs do not appear the same each time the programs are executed, the output almost appears to be in random order for each execution. I speculated that this is because each time the programs are run the scheduled execution time of each thread/process and the actual time it gets on the processor(s) is not guaranteed to be deterministic and is affected by many variables in the system used to test.

2.) What happens when the Pthread version of the program is run when pthread_mutex_lock(&MUTEX) and the pthread_mutex_unlock(&MUTEX) statements are commented out or removed?

(Output)

```
uahcls01@dmcvlogin1:Hw1> ./hello_world_PTH
Hello World, from PThread Hello World, from PThread Hello World, from PThread 213

Hello World, from PThread 5
Hello World, from PThread 4
Number of threads = 8Hello World, from PThread
0Hello World, from PThread 6

Hello World, from PThread 7
```

   a. The output to the terminal is scrambled. This is because without the mutex, the lock that a thread will hold while it works on its task and release when finished, doesn't exist and the threads are free to run when they get the chance. This produces unfinished output to the terminal because many threads are or could be in the middle of executing their section of code.

3.) What happens when the OpenMP version of the program is run when the ***#pragma omp critical*** statements are removed?

(Output)

```
uahcls01@dmcvlogin1:Hw1> ./hello_world_OMP
Hello World from thread = Hello World from thread = 23
Hello World from thread = Hello World from thread = 5
Hello World from thread = Hello World from thread = Hello World from thread = 6
40Hello World from thread =


7
1
Number of Threads = 8
```

a. The output in this case is very similar to the previous case, removing the mutex from the pthreads program, where the output to the terminal is scrambled. This is because the ***#pragma omp critical*** statement is meant to protect against race conditions which in this case would have allowed the cout statements to finish uninterrupted. When taken away we see the intermediate results of the threads as they get time on the processor(s).

Hybrid

HYBRID SOURCE CODE

```
/*
   MPI/OpemMP - Hello World - C++ Version (utilizing C function calling
Conventions)
   FILE: hybrid.cpp

   Compilation on dmc.asc.edu

   first set up environment by typing from the command line

      module load openmpi

   to compile the program type

      mpic++ -o hybrid -fopenmp hybrid.cpp

   to run on eight processors type

      mpirun -np 4 ./hybrid
*/

// Kyle Ray
// CPE_512_Intro_to_Parallel_Programming
// August 31, 2017
// Homework #1
```

```
// Hybrid version utilizing MPI and OpenMP
// Create 4 main MPI processes
// Each process will have two executing threads

using namespace std;
#include <iostream>
#include <mpi.h>
#include <omp.h>

int main (int argc, char *argv[])
{
   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

   // Create the parallel team of two threads for this MPI process
   #pragma omp parallel num_threads(2)
   {
      // Allow the output statements to finish
      #pragma omp critical
      {
         cout << "Hello World from MPI Process #" << rank << " Thread # " <<
omp_get_thread_num() << endl << flush;
      }
   }

   // Only root MPI process does this
   if (rank == 0)
   {
      cout << "Number of MPI Processes = " << nmtsks << endl;
   }

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

}
```

OUTPUT

```
uahcls01@dmcvlogin2:Hw1> mpirun -np 4 ./hybrid
Hello World from MPI Process #1 Thread # 1
Hello World from MPI Process #3 Thread # 1
Hello World from MPI Process #3 Thread # 0
Hello World from MPI Process #1 Thread # 0
Hello World from MPI Process #2 Thread # 1
Hello World from MPI Process #2 Thread # 0
Hello World from MPI Process #0 Thread # 1
Hello World from MPI Process #0 Thread # 0
Number of MPI Processes = 4
```