|  |
| --- |
|  |
| Hello World |
| Homework #1 |

|  |
| --- |
| Kyle Ray  8-31-2017 |

1. Open MPI

Output:

uahcls01@dmcvlogin1:Hw1> module load openmpi

uahcls01@dmcvlogin1:Hw1> mpic++ hello\_world\_MPI.cpp -o hello\_world\_MPI

uahcls01@dmcvlogin1:Hw1> mpirun -np 8 ./hello\_world\_MPI

Hello World from MPI Process #0

Number of MPI Processes = 8

Hello World from MPI Process #6

Hello World from MPI Process #7

Hello World from MPI Process #5

Hello World from MPI Process #1

Hello World from MPI Process #2

Hello World from MPI Process #3

Hello World from MPI Process #4

1. Open MPI 2

Output:

uahcls01@dmcvlogin1:Hw1> mpic++ hello\_world\_MPI2.cpp -o hello\_world\_MPI2

uahcls01@dmcvlogin1:Hw1> mpirun -np 8 ./hello\_world\_MPI2

Hello World from MPI Process #0

Number of MPI Processes = 8

Hello World from MPI Process #6

Hello World from MPI Process #7

Hello World from MPI Process #3

Hello World from MPI Process #5

Hello World from MPI Process #1

Hello World from MPI Process #2

Hello World from MPI Process #4

1. Pthread

Output:

uahcls01@dmcvlogin1:Hw1> g++ hello\_world\_PTH.cpp -o hello\_world\_PTH -lpthread

uahcls01@dmcvlogin1:Hw1> ./hello\_world\_PTH

Hello World, from PThread 1

Hello World, from PThread 0

Hello World, from PThread 3

Hello World, from PThread 2

Hello World, from PThread 4

Hello World, from PThread 5

Hello World, from PThread 6

Number of threads = 8

Hello World, from PThread 7

1. OpenMP

Output:

uahcls01@dmcvlogin1:Hw1> g++ -o hello\_world\_OMP -fopenmp hello\_world\_OMP.cpp

uahcls01@dmcvlogin1:Hw1> ./hello\_world\_OMP

Hello World from thread = 5

Hello World from thread = 3

Hello World from thread = 1

Hello World from thread = 6

Hello World from thread = 7

Hello World from thread = 2

Hello World from thread = 4

Hello World from thread = 0

Number of Threads = 8

# Answers

1. Do the outputs always appear the same each time the program is executed? **(NEEDS REWORKED)**
   1. The outputs do not appear the same each time the programs are executed. I speculate that this is because of the scheduled execution time of the thread and when it actually gets time on a core. I’m not too familiar with how OpenMPI, Pthreads, and OpenMP perform this parallelism.
2. What happens when the Pthread version of the program is run when the pthread\_mutex\_lock(&MUTEX) and the pthread\_mutex\_unlock(&MUTEX) statemets are commented out or removed? (**NEEDS REWORKED)**

uahcls01@dmcvlogin1:Hw1> vim hello\_world\_PTH.cpp

uahcls01@dmcvlogin1:Hw1> g++ hello\_world\_PTH.cpp -o hello\_world\_PTH -lpthread

uahcls01@dmcvlogin1:Hw1> ./hello\_world\_PTH

Hello World, from PThread Hello World, from PThread 1Hello World, from PThread

Hello World, from PThread 5

Hello World, from PThread 4

Hello World, from PThread 2

Hello World, from PThread 6

0

Number of threads = 8

3

Hello World, from PThread 7

1. The output is scrambled. This is caused by the lack of locking/unclocking the mutual exclusion lock; therefore, the processors are fighting against each other for resources.
2. What happens when the OpenMP version of the program is run when the #pragma omp critical statements are removed?

uahcls01@dmcvlogin1:Hw1> vim hello\_world\_OMP.cpp

uahcls01@dmcvlogin1:Hw1> g++ -o hello\_world\_OMP -fopenmp hello\_world\_OMP.cpp

uahcls01@dmcvlogin1:Hw1> ./hello\_world\_OMP

Hello World from thread = Hello World from thread = Hello World from thread = Hello World from thread = 7034Hello World from thread =

Hello World from thread = Hello World from thread = Hello World from thread = 1

5

Number of Threads = 8

6

2

* 1. The output is scrambled as well. This is because the #pragma omp critical statement is meant to protect against race conditions. It serializes the execution and allowed the cout statements to finish butwhen we take that away we are not guaranteed this safety net.