

Homework #1

cpe 512

Kyle Ray

August 31, 2017

Contents

[Example Program Output 2](#_Toc491894524)

[MPI 2](#_Toc491894525)

[MPI2 2](#_Toc491894526)

[Pthread 2](#_Toc491894527)

[OpenMP 2](#_Toc491894528)

[Questions 3](#_Toc491894529)

[Hybrid 3](#_Toc491894530)

[Hybrid Source Code 3](#_Toc491894531)

[Output 5](#_Toc491894532)

[Appendix 6](#_Toc491894533)

# 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?
   1. The outputs do not appear the same each time the programs are executed. This is because each time the programs are run the scheduled execution time of each thread and the actual time it gets on the processor(s) is not guaranteed to be deterministic.
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?
   1. **(Output)**
   2. 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?
   1. **(Output)**
   2. 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. It serializes the execution and in this case, allows 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



# Appendix

Add anything else that might be pertinent to the assignment.