Parallel and Distributed Computing

CSE4001

Fall Semester 2020-21

Lab Assignment 6

**ISHAAN OHRI**

**18BCE0265**

**Aim:**

Consider a suitable instance that has MPI routines to assign different tasks to different processors.

For example, parts of an input data set might be divided and processed by different processors, or a finite difference grid might be divided among the processors available. This means that the code needs to identify processors. In this example, processors are identified by rank - an integer from 0 to total number of processors.

1. Implement the logic using C
2. Build the code
3. Show the screenshots with proper justification

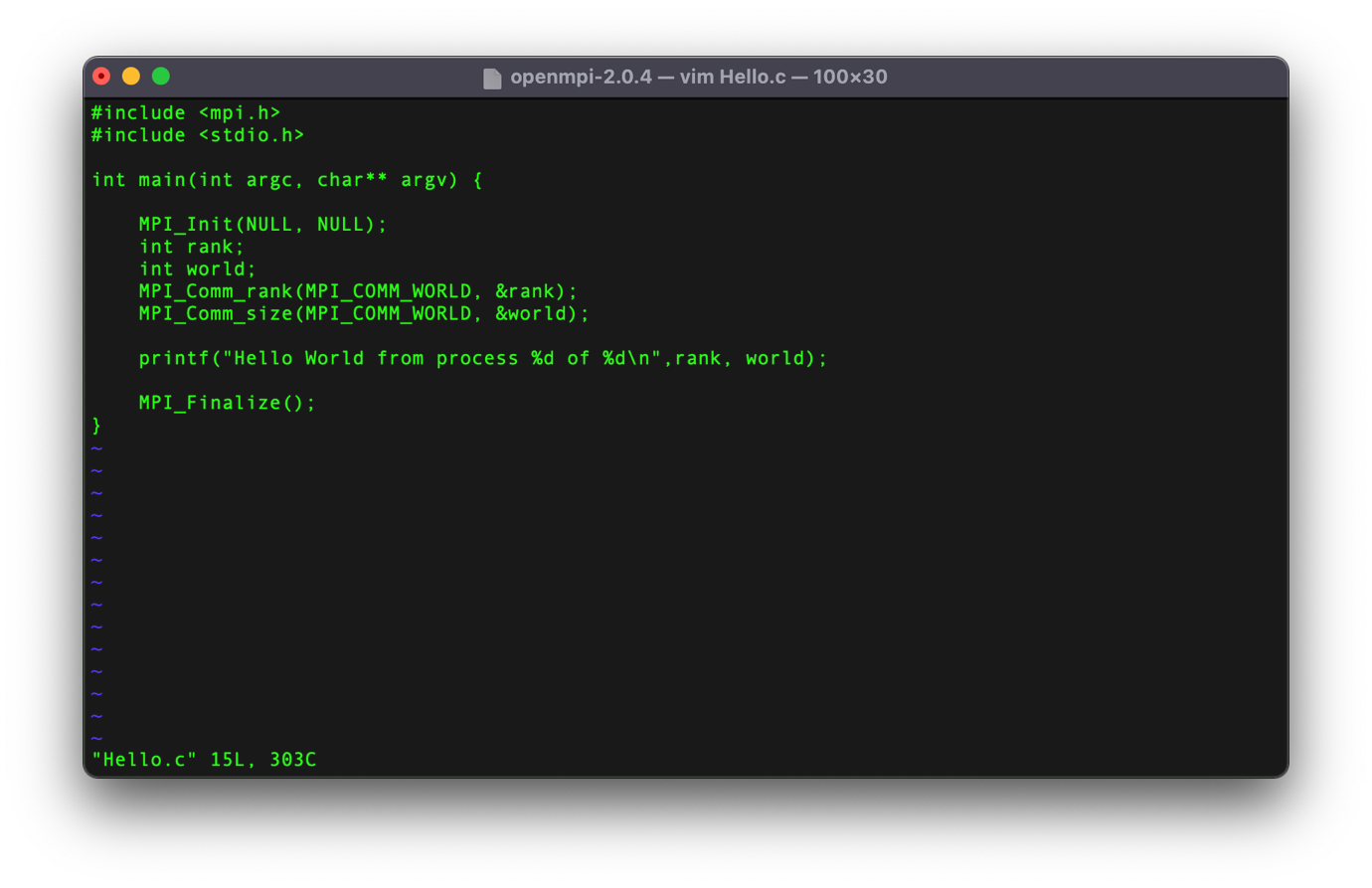
**Note.**

**Compile and run with:**

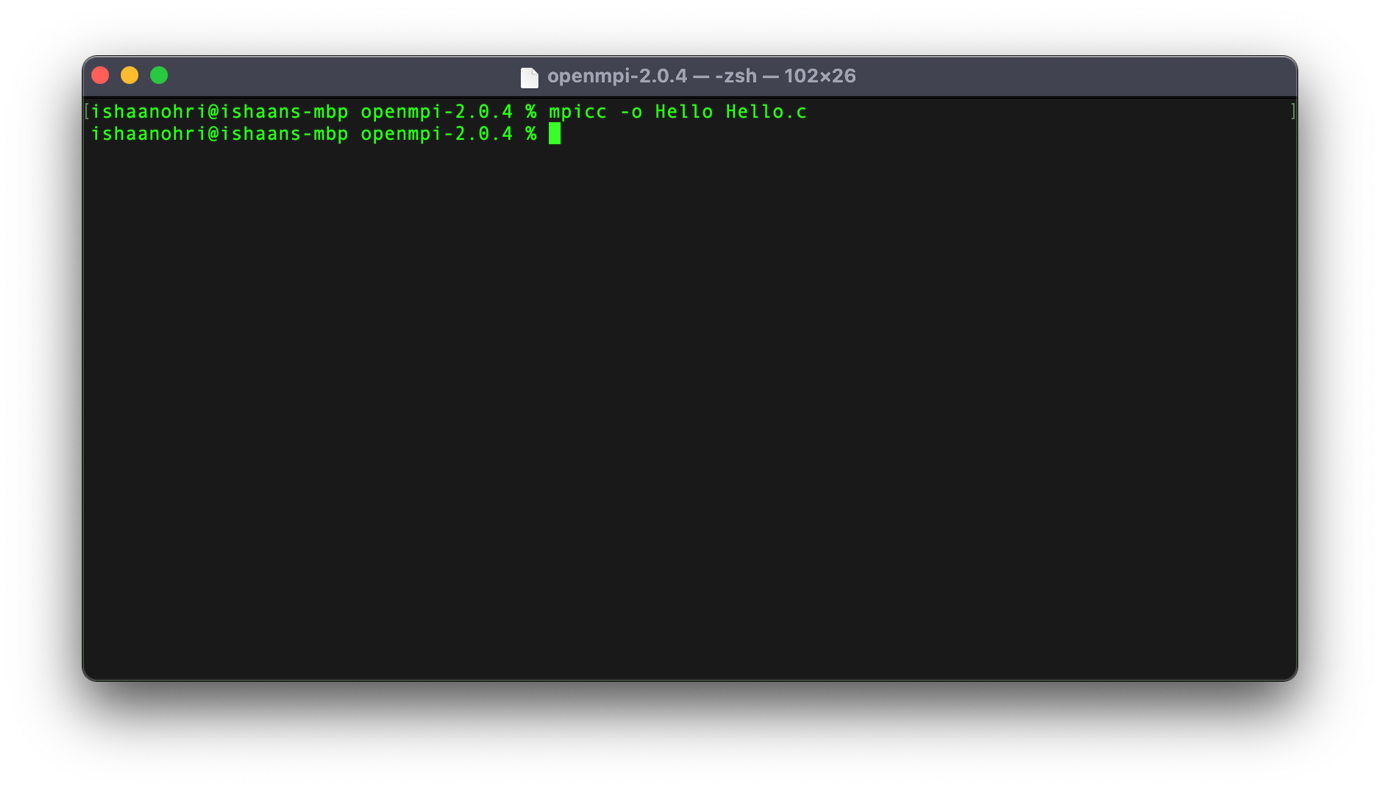
mpicc -o Hello Hello.c

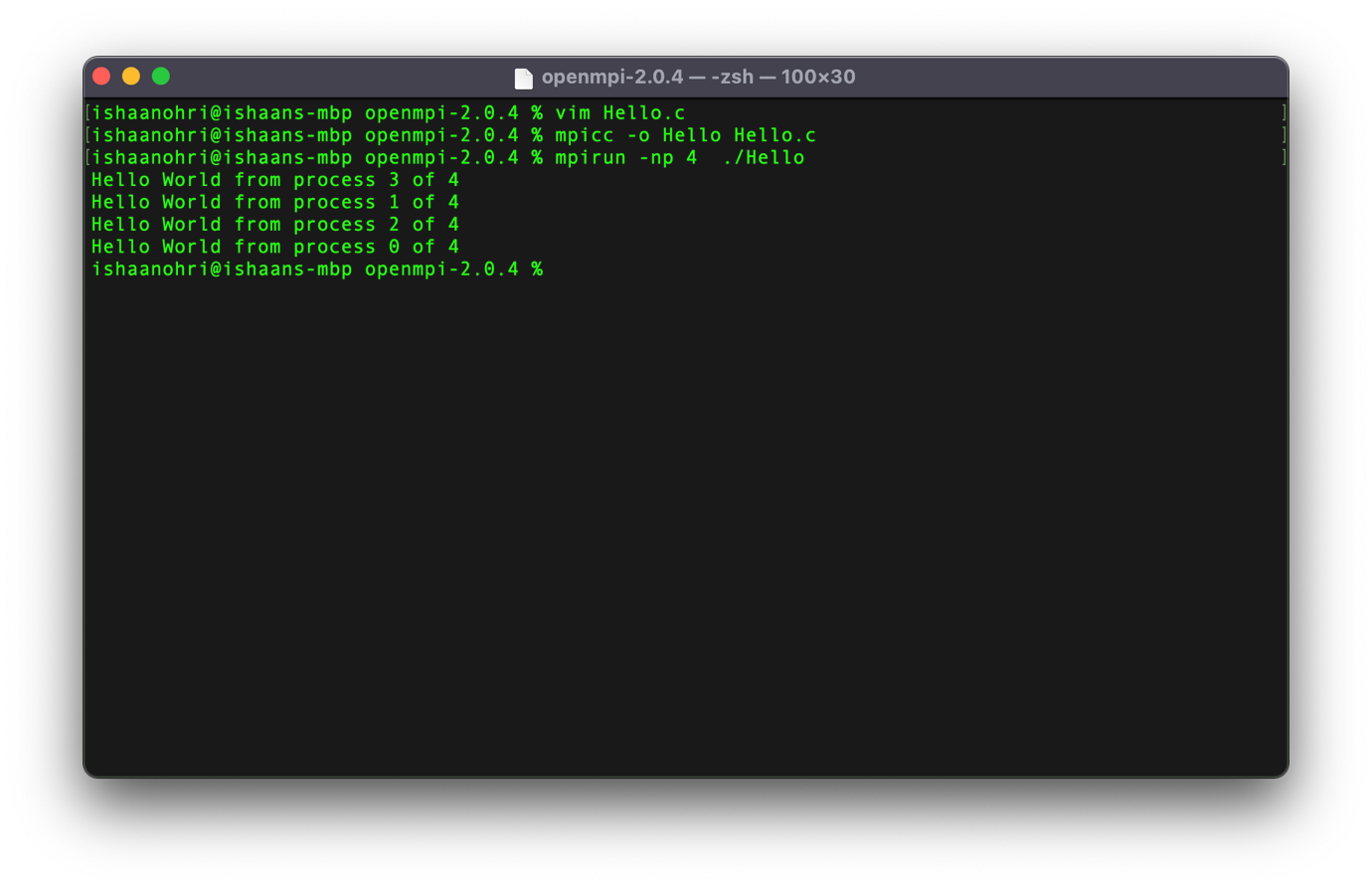
mpirun -np 4 ./Hello

**Source Code:**

****

**Execution:**

****



**Result:**

From this experiment, after implementing the program I learnt the following:

The command MPI\_Init(NULL, NULL) initializes the MPI execution environment with the two arguments being pointer to the number of arguments and the pointer to the argument vector.

The command MPI\_Comm\_rank() is used to determine the rank of the calling process in the communicator. It takes the communicator as an argument.

The command MPI\_Comm\_size() is used to determine the size of the calling process in the communicator. It takes the communicator as an argument.

The command MPI\_Finalize() is used to terminate the MPI execution environment. All processes must call this routine before exiting. The number of processes running after this routine is called is undefined.

The above experiment was conducted and all results along with the source code have been attached above in the document. The experiment was assisted by Dr Deebak. I thank sir for his assistance.