Computer Science & Engineering

CSE4001

Parallel and Distributed Computing

LAB ASSIGNMENT 6

Submitted to **Prof. DEEBAK B.D.**

TOPIC: PROBLEMS USING MPI

NAME: PUNIT MIDDHA

REG.NO: 19BCE2060

SLOT: L55+L56

DATE: 20/10/2021

QUESTION – I

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

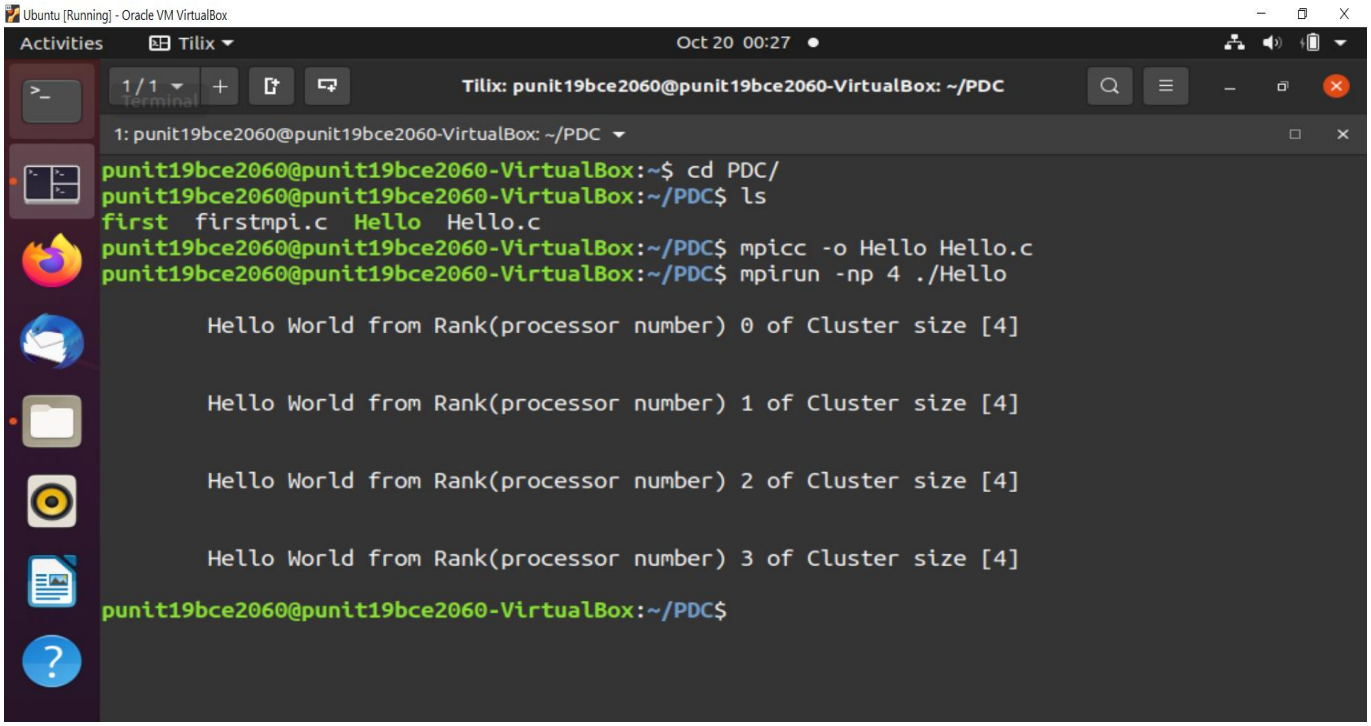
mpirun -np 4 ./Hello

SOURCE CODE:

```
#include<stdio.h>

#include<mpi.h>

#include<stdlib.h>


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

int Process_Rank, Cluster_Size;

MPI_Init(&argc, &argv);

MPI_Comm_rank(MPI_COMM_WORLD, &Process_Rank);

MPI_Comm_size(MPI_COMM_WORLD, &Cluster_Size);

printf("\n\tHello World from Rank(processor number) %d of Cluster size
[%d]\n", Process_Rank, Cluster_Size);

printf("\n");

MPI_Finalize();

return 0;

}
```


```c
#include<stdio.h>
#include<mpi.h>
#include<stdlib.h>

int main(int argc, char **argv){
        int Process_Rank, Cluster_Size;
        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD, &Process_Rank);
        MPI_Comm_size(MPI_COMM_WORLD, &Cluster_Size);
        printf("\n\tHello World from Rank(processor number) %d of Cluster size [%d]\n", Process_Rank, Cluster_Size);
        printf("\n");
        MPI_Finalize();
        return 0;
}
```

EXECUTION:



REMARKS:

Despite the fact that MPI is a complex and multifaceted system, we can solve a wide range of problems by utilising only six of its functions! We begin by describing the six functions that initiate and terminate computations, identify processes, and send and receive messages in MPI:

- MPI_INIT is used to start an MPI computation.
- MPI_FINALIZE: This function ends a computation.
- MPI_COMM_SIZE: Count the number of processes.
- MPI_COMM_RANK: Find my process identifier.
- MPI_COMM_WORLD: MPI COMM WORLD communicates. All MPI communication calls require a communicator argument, and MPI processes can only communicate with one another if they share a communicator.