Fall Sem 2021-22

Assignment: 7

Date: 10/11/21

Name: Ishan Sagar Jogalekar

Reg no: 19BCE2250

Course: Parallel and distributed computing LAB - CSE4001

Slot: L55+L56

Aim: Consider the following program, called mpi_sample1.c. This program is written in C with MPI commands included.

The new MPI calls are to MPI_Send and MPI_Recv and to MPI_Get_processor_name. The latter is a convenient way to get the name of the processor on which a process is running. MPI_Send and MPI_Recv can be understood by stepping back and considering the two requirements that must be satisfied to communicate data between two processes:

1. Describe the data to be sent or the location in which to receive the data

2. Describe the destination (for a send) or the source (for a receive) of the data.

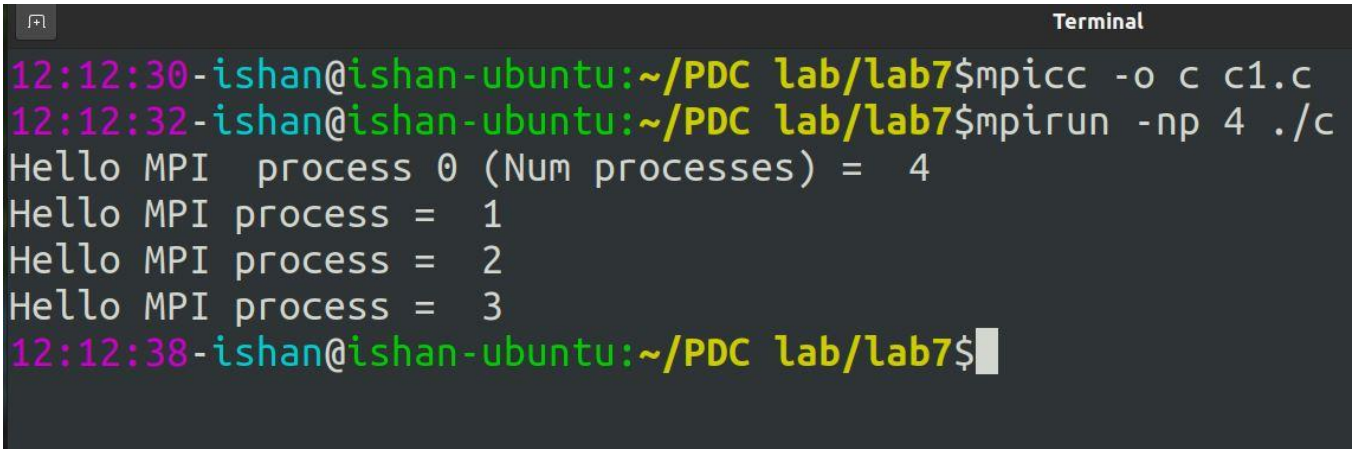
Ans –

SOURCE CODE:

```
#include <stdio.h>
#include <string.h>
#include <mpi.h>

int main(int argc,char* argv[])
{
   int my_rank; /* rank of process */
   int p; /* number of processes */
   int source; /* rank of sender */
   int dest; /* rank of receiver */
   int tag=0; /* tag for messages */
   char message[100]; /* storage for message */
   MPI_Status status ; /* return status for receive */
```

```
    /* start up MPI */
    MPI_Init(&argc,&argv);

    /* find out process rank */
    MPI_Comm_rank(MPI_COMM_WORLD,&my_rank);

    /*find out number of processes*/
    MPI_Comm_size(MPI_COMM_WORLD,&p);
    if(my_rank!=0)
    {
       /*create message*/
       sprintf(message,"Hello MPI process =  %d ",my_rank);
       dest=0;
       /*use strlen+1 so that '\0' get transmitted*/

MPI_Send(message,strlen(message)+1,MPI_CHAR,dest,tag,MPI_COMM_WOR
LD);
    }
    else
    {
       printf("Hello MPI  process 0 (Num processes) =  %d\n",p);
       for(source=1;source<p;source++)
       {

MPI_Recv(message,sizeof(message),MPI_CHAR,source,tag,MPI_COMM_WORL
D,&status);
            printf("%s\n",message);
       }
    }
    /* shut down MPI */
    MPI_Finalize();
    return 0 ;
}
```

EXECUTION:



RESULTS:

- OpenMP is a library for parallel programming in the SMP (symmetric multi-processors or shared-memory processors) model.
- Mpi.h is header file to use all mpi functions inside program.
- For current program 6 mpi functions are used which initiate, terminate computation, identify process rank and send, receive message between processors.
- 6 functions:
    1. MPI_INIT : Initiate an MPI computation.
    2. MPI_FINALIZE : Terminate a computation.
    3. MPI_COMM_SIZE : Determine number of processes.
    4. MPI_COMM_RANK : Determine my process identifier.
    5. MPI_SEND : Send a message.
    6. MPI_RECV : Receive a message.