CSE4001 - Parallel and Distributed Computing, Fall 2019
Vellore Institute of Technology
Instructor: Prof Deebak B D - SCOPE

Lab report

Title of Lab: Beginning with MPI
Assessment #: 9
Date: 04|10|2019

Author's name: Gagan Deep Singh
Registration ID: 17BCI0140
Lab section: Friday L59 + L60

AIM: Implement the given code in Ubuntu. Build and Execute Its Logical Scenario. Depict the screenshots along with proper justification.

SOURCE CODE:

```
#include <stdio.h>
#include <string.h>
#include "mpi.h"
int main(int argc, char* argv[]){
        int my_rank, p, source, dest, tag=0;
        char message[100];
        MPI_Status status;
        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
        MPI_Comm_size(MPI_COMM_WORLD, &p);
        if (my_rank !=0){
                sprintf(message, "Hello MPI World from process %d!", my_rank);
                dest = 0;
                MPI_Send(message, strlen(message)+1, MPI_CHAR, dest, tag,
MPI_COMM_WORLD);
        }
        else{
                printf("Hello MPI World From process 0: Num processes: %d\n",p);
                for(source = 1; source < p; source++){
                        MPI_Recv(message, 100, MPI_CHAR, source, tag,
MPI_COMM_WORLD, &status);
                        printf("%s\n",message);
                }
        }
        MPI_Finalize();
        return 0;
}
```

EXECUTION:

Code Writer

PDC17BCI0140.c

```c
#include <stdio.h>
#include <string.h>
#include "mpi.h"
int main(int argc, char* argv[]){
    int my_rank, p, source, dest, tag=0;
    char message[100];
    MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank)
    MPI_Comm_size(MPI_COMM_WORLD, &p);
    if (my_rank !=0){
        sprintf(message, "Hello MPI World f
        dest = 0;
        MPI_Send(message, strlen(message)+1
    }
    else{
        printf("Hello MPI World From proces
        for(source = 1; source < p; source+
            MPI_Recv(message, 100, MPI_CHAR
            printf("%s\n",message);
        }
    }
    MPI_Finalize();
    return 0;
}
```

gagandeep@GAGAN: /mnt/e

```
gagandeep@GAGAN:/mnt/e$ lamboot

LAM 7.1.4/MPI 2 C++/ROMIO - Indiana University

gagandeep@GAGAN:/mnt/e$ mpirun -np 4 ./A
Hello MPI World From process 0: Num processes: 4
Hello MPI World from process 1!
Hello MPI World from process 2!
Hello MPI World from process 3!
gagandeep@GAGAN:/mnt/e$
```

The new MPI calls are to MPI_Send, MPI_Recv and MPI_Get_processor_name (this gets the name of the processor on which a process is running). For MPI_Send and MPI_Recv there are 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.