Fall Sem 2021-22

Assignment: 8

Date: 23/11/21

Name: Ishan Sagar Jogalekar

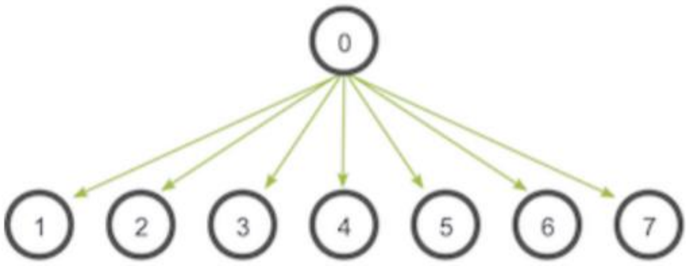
Reg no: 19BCE2250

Course: Parallel and distributed computing LAB - CSE4001

Slot: L55+L56

Aim:

Write a ‘C’ program to initialize the communication pattern of a broadcast. The code logic can typically have a process zero [as root], which has the initial copy of the data to broadcast to other processes [as shown in the below figure].



Ans –

SOURCE CODE:

```
#include <mpi.h>

#include <stdio.h>

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

        //19BCE2250- Ishan Jogalekar


        int rank;

        char buf[50] = "Hello";
```

```
    const int root=0;
   //MPI starting and also finding rank for each process
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    //Message call for root (0th Process)
    if(rank == root) {
        printf("\n[%d]: Before Bcast, Message: %s\n\n", rank, buf);
    }
    //everyone calls bcast, data is taken from root and ends up in everyone's buf
     //MPI Bcast functiom
    MPI_Bcast(&buf, 1, MPI_CHAR, root, MPI_COMM_WORLD);

    //Printing after Bcast function
    printf("[%d]: After Bcast,Message: %s\n\n", rank, buf);

    //Ending MPI
    MPI_Finalize();
    return 0;
}
```

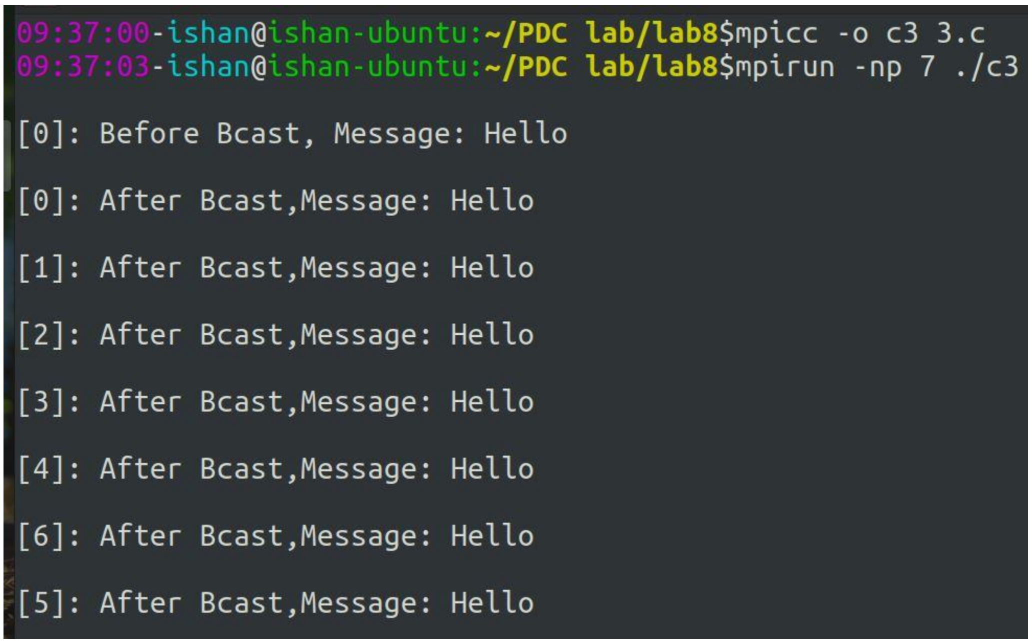
EXECUTION:

1. OpenMP is a library for parallel programming in the SMP (symmetric multi-processors or shared-memory processors) model.
2. Mpi.h is header file to use all mpi functions inside program.
3. MPI_COMM_RANK is used to determine process identifier, that processes id number.
4. MPI_Bcast is used to broadcast the message from one root source to multiple process.

5. If statement is used to check root using rank of process when conditions fail then root sends or broadcast to other 7 processors.
6. Finally, it prints after Broadcasting message with rank of process per processor.
7. While using MPI_Bcast function setting data-type MPI_CHAR as message is “Hello” as character array or string.

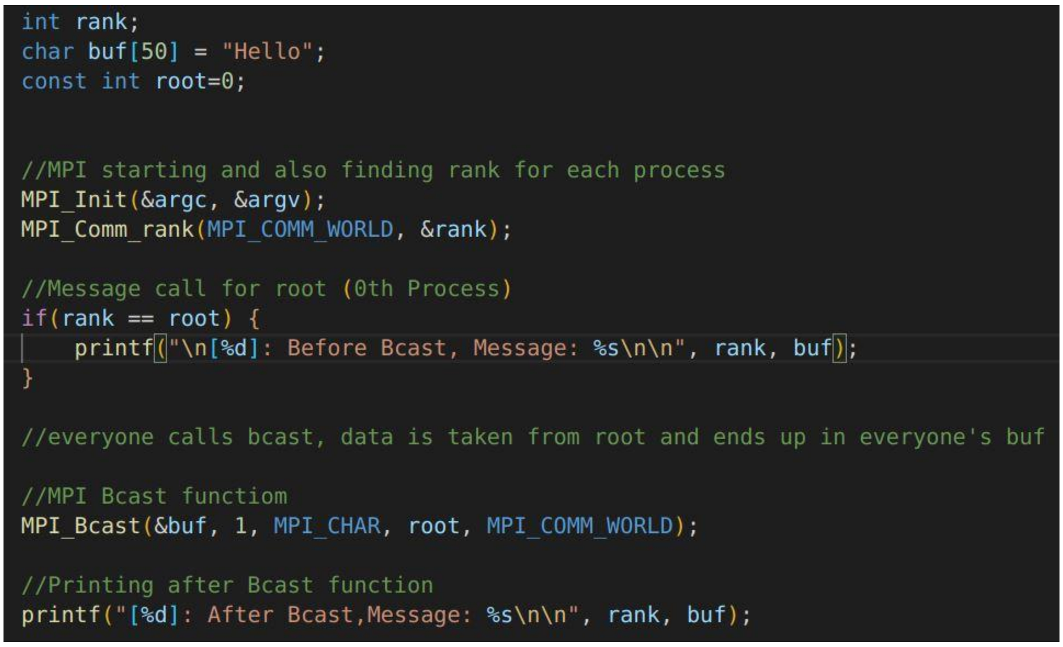
RESULTS:

Output:



```
09:37:00-ishan@ishan-ubuntu:~/PDC lab/lab8$mpicc -o c3 3.c
09:37:03-ishan@ishan-ubuntu:~/PDC lab/lab8$mpirun -np 7 ./c3

[0]: Before Bcast, Message: Hello

[0]: After Bcast,Message: Hello

[1]: After Bcast,Message: Hello

[2]: After Bcast,Message: Hello

[3]: After Bcast,Message: Hello

[4]: After Bcast,Message: Hello

[6]: After Bcast,Message: Hello

[5]: After Bcast,Message: Hello
```

Code:



```c
int rank;
char buf[50] = "Hello";
const int root=0;


//MPI starting and also finding rank for each process
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);

//Message call for root (0th Process)
if(rank == root) {
    printf("\n[%d]: Before Bcast, Message: %s\n\n", rank, buf);
}

//everyone calls bcast, data is taken from root and ends up in everyone's buf

//MPI Bcast function
MPI_Bcast(&buf, 1, MPI_CHAR, root, MPI_COMM_WORLD);

//Printing after Bcast function
printf("[%d]: After Bcast,Message: %s\n\n", rank, buf);
```