



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Computer Science & Engineering

CSE4001

Parallel and Distributed Computing

LAB ASSIGNMENT 8

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

TOPIC: PROBLEMS USING MPI

NAME: PUNIT MIDDHA

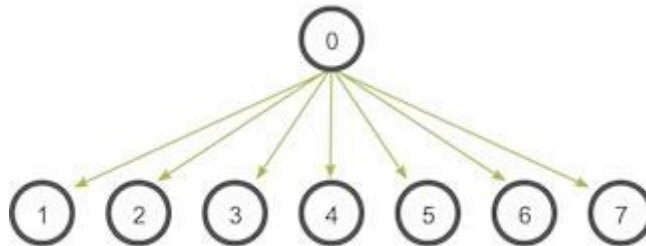
REG.NO: 19BCE2060

SLOT: L55+L56

DATE: 23/11/2021

QUESTION – I

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].



Hint: The function prototype is as follows:

MPI_Bcast(

void* data,

int count,

MPI_Datatype datatype,

int root,

MPI_Comm communicator)

SOURCE CODE:

```
#include <stdio.h>
#include<stdlib.h>
#include <mpi.h>
int main(int argc, char** argv) {
//Initailize MPI and finding the rank for each process
    MPI_Init(&argc, &argv);
    int process_rank;
    MPI_Comm_rank(MPI_COMM_WORLD,&process_rank);

    char message[30] = "Hello World! REGNO: 19BCE2060";
    const int bcast_root=0;

    //Call for root
    if(process_rank == bcast_root) {
        printf("\n[Process Number: %d] From Broadcast Root, Send Message: %s\n\n",
            process_rank, message);
    }

    //MPI Bcast function
```

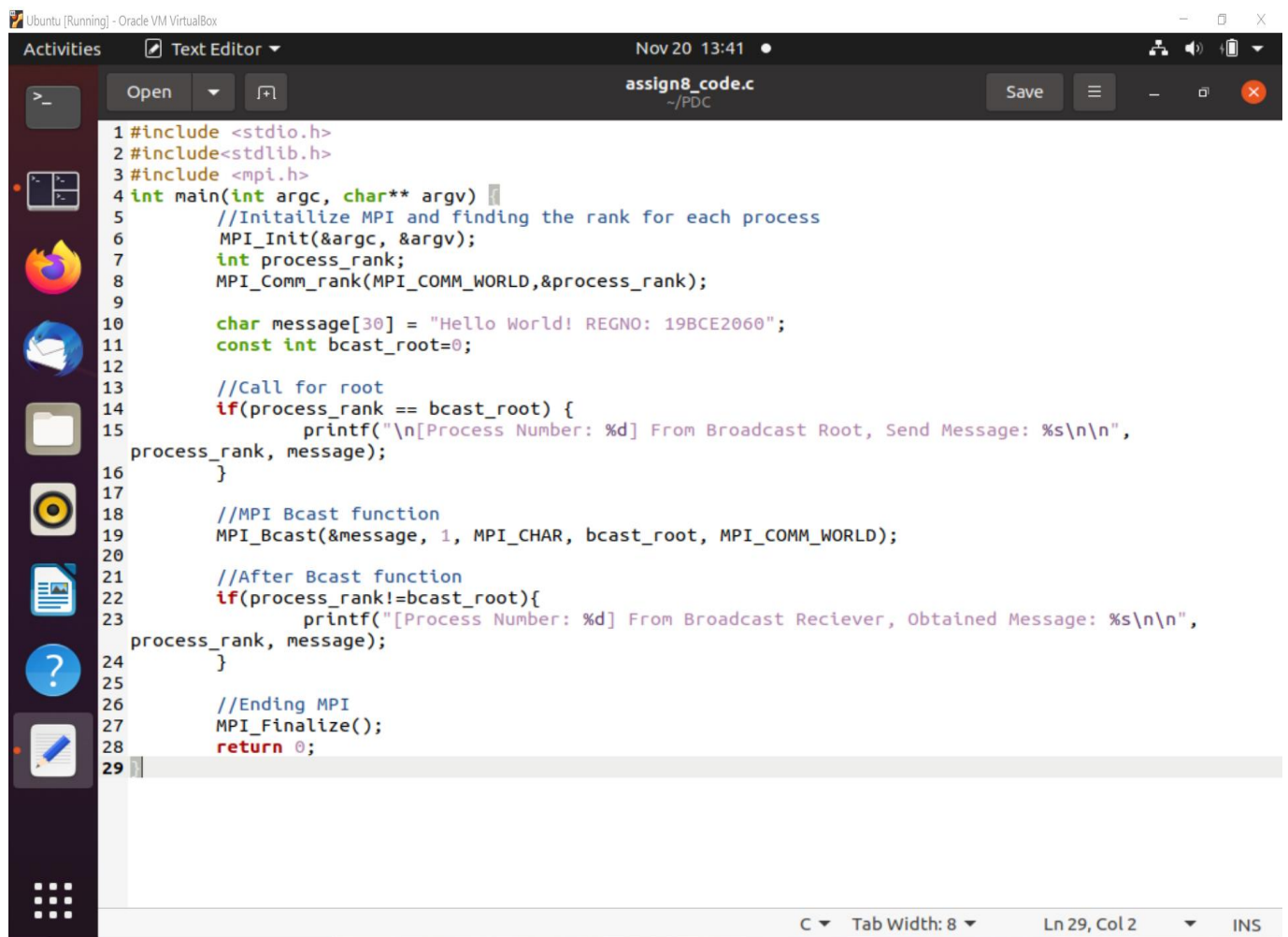
```
MPI_Bcast(&message, 1, MPI_CHAR, bcast_root, MPI_COMM_WORLD);
```

```
//After Bcast function
```

```
if(process_rank!=bcast_root){  
printf("[Process Number: %d] From Broadcast Reciever, Obtained Message:  
%s\n\n", process_rank, message);  
}
```

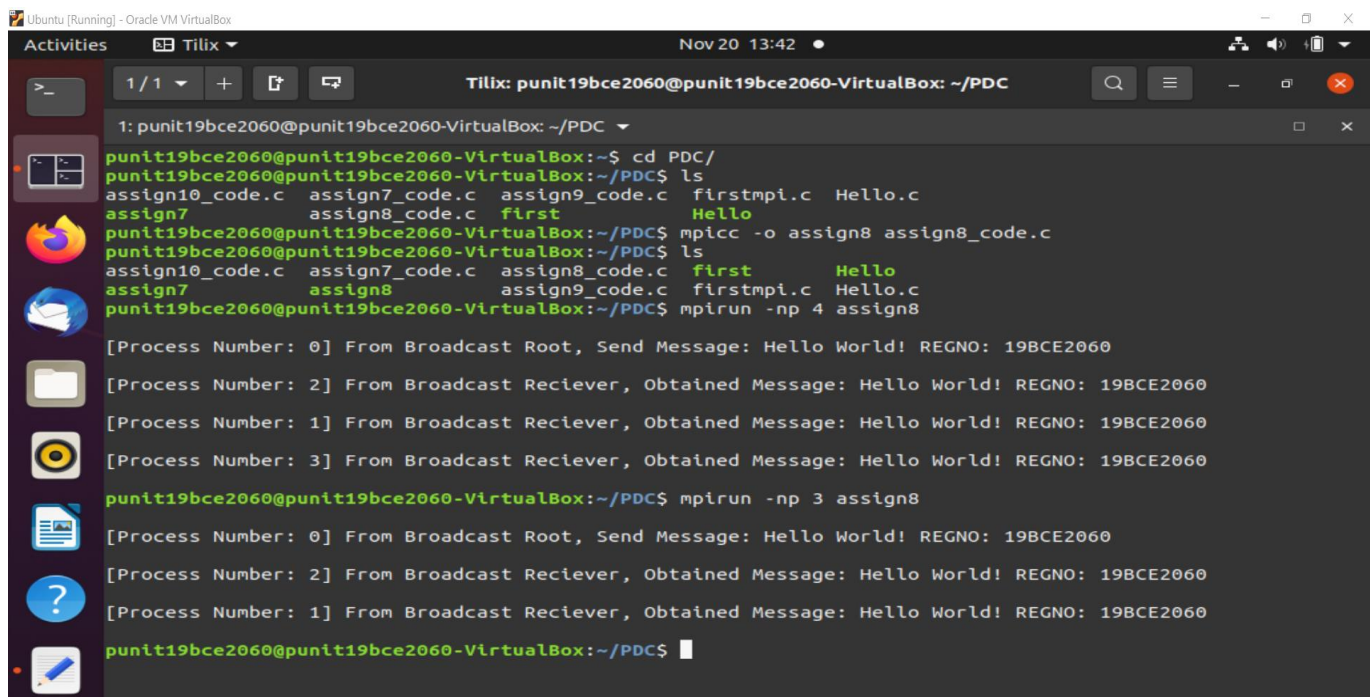
```
//Ending MPI
```

```
MPI_Finalize();  
return 0;  
}
```

A screenshot of a code editor window titled 'assign8_code.c' in a dark theme. The editor shows the C code for an MPI broadcast program. The code includes headers for stdio, stdlib, and mpi. The main function initializes MPI, finds the process rank, and sets a message. It then calls MPI_Bcast. After the broadcast, it checks if the process is not the root and prints the received message. Finally, it calls MPI_Finalize and returns 0. The status bar at the bottom indicates 'Ln 29, Col 2' and 'INS' mode.

```
1 #include <stdio.h>  
2 #include<stdlib.h>  
3 #include <mpi.h>  
4 int main(int argc, char** argv) {  
5     //Initailize MPI and finding the rank for each process  
6     MPI_Init(&argc, &argv);  
7     int process_rank;  
8     MPI_Comm_rank(MPI_COMM_WORLD,&process_rank);  
9  
10    char message[30] = "Hello World! REGNO: 19BCE2060";  
11    const int bcast_root=0;  
12  
13    //Call for root  
14    if(process_rank == bcast_root) {  
15        printf("\n[Process Number: %d] From Broadcast Root, Send Message: %s\n\n",  
process_rank, message);  
16    }  
17  
18    //MPI Bcast function  
19    MPI_Bcast(&message, 1, MPI_CHAR, bcast_root, MPI_COMM_WORLD);  
20  
21    //After Bcast function  
22    if(process_rank!=bcast_root){  
23        printf("[Process Number: %d] From Broadcast Reciever, Obtained Message: %s\n\n",  
process_rank, message);  
24    }  
25  
26    //Ending MPI  
27    MPI_Finalize();  
28    return 0;  
29 }
```

EXECUTION:



The screenshot shows a terminal window titled 'Tilix: punit19bce2060@punit19bce2060-VirtualBox: ~/PDC'. The user is in the directory ~/PDC. The terminal output shows the following commands and results:

```
punit19bce2060@punit19bce2060-VirtualBox:~$ cd PDC/
punit19bce2060@punit19bce2060-VirtualBox:~/PDC$ ls
assign10_code.c  assign7_code.c  assign9_code.c  firstmpi.c  Hello.c
assign7          assign8_code.c  first           Hello
punit19bce2060@punit19bce2060-VirtualBox:~/PDC$ mpicc -o assign8 assign8_code.c
punit19bce2060@punit19bce2060-VirtualBox:~/PDC$ ls
assign10_code.c  assign7_code.c  assign8_code.c  first           Hello
assign7          assign8         assign9_code.c  firstmpi.c     Hello.c
punit19bce2060@punit19bce2060-VirtualBox:~/PDC$ mpirun -np 4 assign8

[Process Number: 0] From Broadcast Root, Send Message: Hello World! REGNO: 19BCE2060
[Process Number: 2] From Broadcast Reciever, Obtained Message: Hello World! REGNO: 19BCE2060
[Process Number: 1] From Broadcast Reciever, Obtained Message: Hello World! REGNO: 19BCE2060
[Process Number: 3] From Broadcast Reciever, Obtained Message: Hello World! REGNO: 19BCE2060
punit19bce2060@punit19bce2060-VirtualBox:~/PDC$ mpirun -np 3 assign8
[Process Number: 0] From Broadcast Root, Send Message: Hello World! REGNO: 19BCE2060
[Process Number: 2] From Broadcast Reciever, Obtained Message: Hello World! REGNO: 19BCE2060
[Process Number: 1] From Broadcast Reciever, Obtained Message: Hello World! REGNO: 19BCE2060
punit19bce2060@punit19bce2060-VirtualBox:~/PDC$
```

REMARKS:

- ✓ There are four processes in total, with the bcast root set to process 0.
- ✓ When the process rank is 0, a message "Hello World! REGNO: 19BCE2060" is sent from root to all other processes 1, 2, and 3.
- ✓ This message is broadcasted using the MPI Bcast function.
- ✓ It sends a message to all other communicator processes from the process with the rank "root," i.e., process 0 in this case.
- ✓ We can observe that the message is sent from process 0 to processes 1, 2, and 3 during execution.