EXPERIMENT NO. 2

Title: To develop a distributed application using Message Passing Interface (MPI)

Objective:

To design a distributed client–server application using MPI for remote computation where the client submits a string to the server, and the server returns the reverse of it.

💡 Theory Recap

MPI (Message Passing Interface) is a standardized and portable message-passing system designed to allow parallel and distributed computing through explicit process communication.

Each process runs independently and exchanges messages using MPI_Send and MPI_Recv.

💻 Executable Program

🔹 Program: String Reversal using MPI (Client-Server Model)

```
// Filename: mpi_string_reverse.c
#include <mpi.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
   int rank, size;
   char str[100];
   MPI_Status status;

   // Initialize MPI environment
   MPI_Init(&argc, &argv);

   // Get current process rank and total number of processes
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);

   if (size < 2) {
      if (rank == 0)
         printf("Please run with at least 2 processes.\n");
      MPI_Finalize();
      return 0;
   }

   if (rank == 0) {
      // Client process
      printf("CLIENT (Process %d): Enter a string: ", rank);
      fflush(stdout);
      fgets(str, sizeof(str), stdin);
      str[strcspn(str, "\n")] = '\0'; // Remove newline

      // Send string to server
      MPI_Send(str, strlen(str) + 1, MPI_CHAR, 1, 0, MPI_COMM_WORLD);

      // Receive reversed string from server
```

```
        MPI_Recv(str, 100, MPI_CHAR, 1, 1, MPI_COMM_WORLD, &status);
        printf("CLIENT: Received reversed string from SERVER: %s\n", str);
    }
    else if (rank == 1) {
        // Server process
        MPI_Recv(str, 100, MPI_CHAR, 0, 0, MPI_COMM_WORLD, &status);

        printf("SERVER (Process %d): Received string: %s\n", rank, str);

        // Reverse the string
        int len = strlen(str);
        for (int i = 0; i < len / 2; i++) {
            char temp = str[i];
            str[i] = str[len - i - 1];
            str[len - i - 1] = temp;
        }

        // Send reversed string back to client
        MPI_Send(str, strlen(str) + 1, MPI_CHAR, 0, 1, MPI_COMM_WORLD);
    }

    // Finalize MPI environment
    MPI_Finalize();
    return 0;
}
```

⚙️ Compilation and Execution Steps

Step 1: Save the program

Save it as:

mpi_string_reverse.c

Step 2: Compile using MPICC

```
mpicc mpi_string_reverse.c -o mpi_string_reverse
```

Step 3: Run the program using two processes

```
mpirun -np 2 ./mpi_string_reverse
```

📄 Sample Output

Client terminal:

```
CLIENT (Process 0): Enter a string: HelloMPI
SERVER (Process 1): Received string: HelloMPI
CLIENT: Received reversed string from SERVER: IPMolleH
```

🧩 Explanation

Role	MPI Function	Purpose
Client	MPI_Send()	Sends the string to the server
Server	MPI_Recv()	Receives the string from client
Server	Reverses string	Performs computation
Server	MPI_Send()	Sends reversed string back
Client	MPI_Recv()	Receives reversed string

🧠 Conclusion
We successfully implemented a client–server communication model using MPI.
The client sends a string to the server, which reverses it and returns it to the client.
This demonstrates message passing, synchronization, and data exchange between distributed processes in