Final Year B.Tech. (CSE) – VII [2024-25]

**6CS452: High Performance Computing Lab**

Assignment No: 8

**Implementation of MPI programs**

**PRN:** 21510038  **Name:** Aniket Raju Ghotkar

**Title:** Implementation of MPI programs

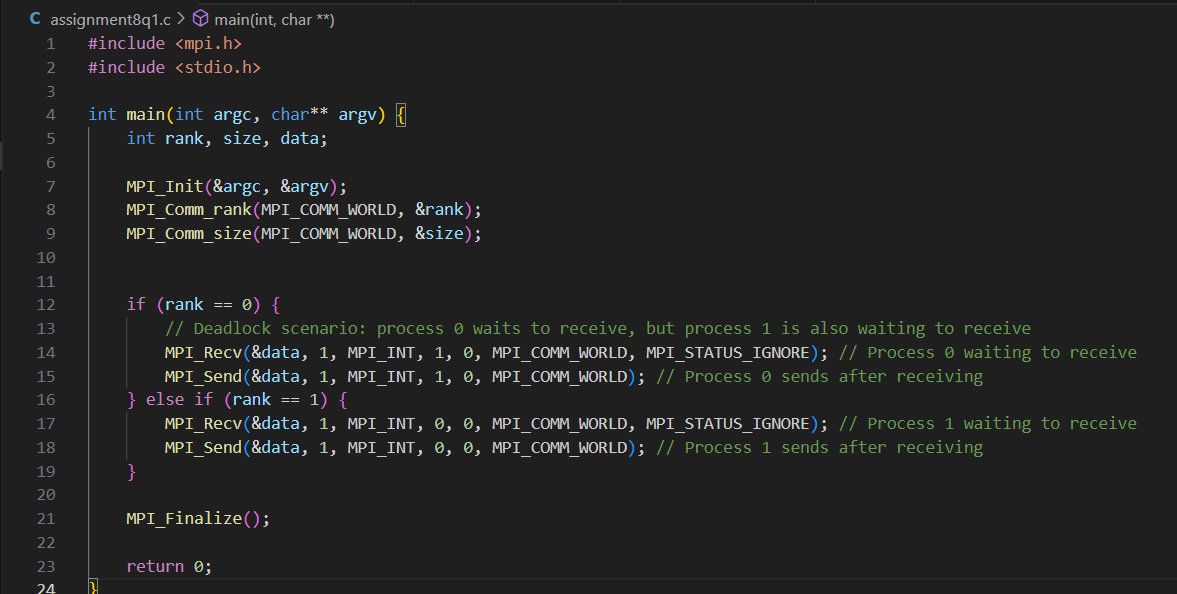
**Q1.** **Implement a MPI program to give an example of Deadlock.**

**Ans:**

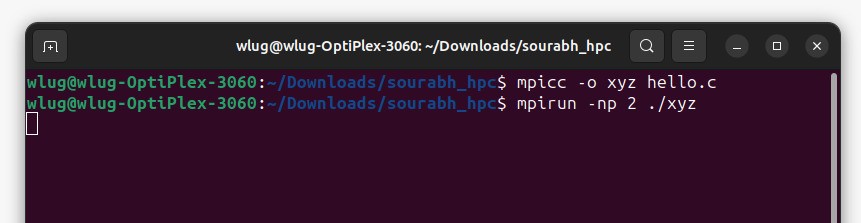
MPI Program Demonstrating Deadlock

Deadlock occurs when two or more processes are blocked because each is waiting for the other to release a resource. In MPI, this can happen if two processes are waiting for each other to send or receive data.

Screenshots:



The program will likely hang due to deadlock.

****

Explaination/Information:

Both processes P0 and P1 wait for each other to send data, resulting in a deadlock because no process can proceed without receiving data first.

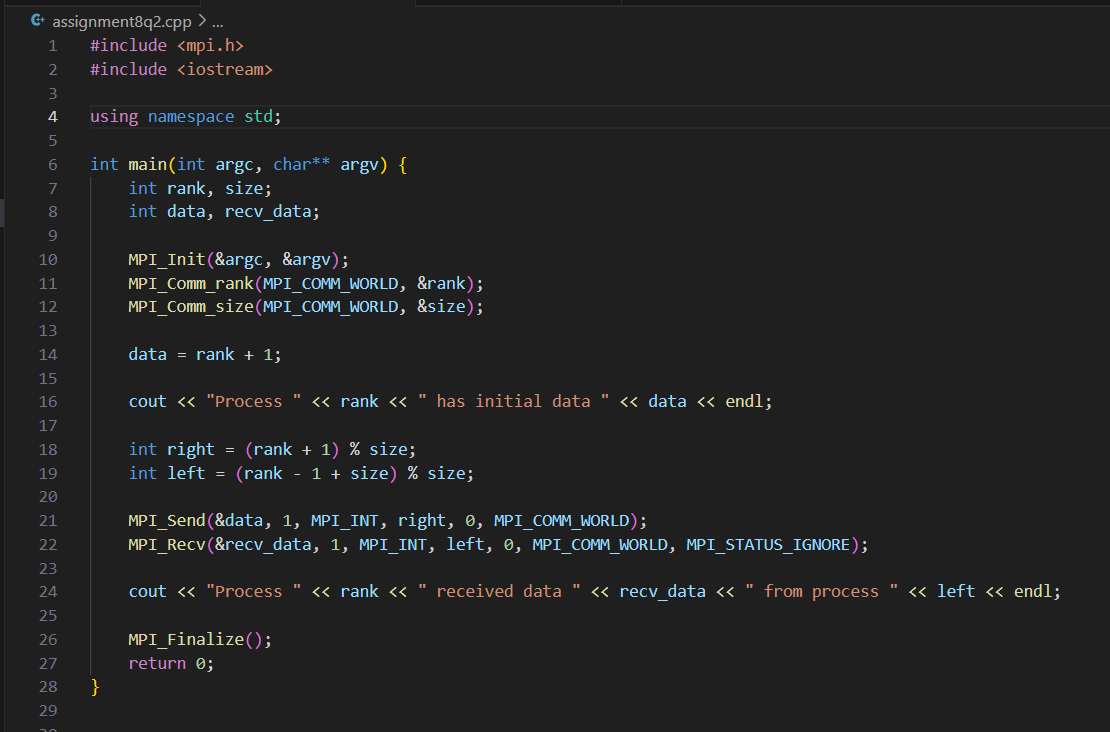
**Q2.** Implement blocking MPI send & receive to demonstrate Nearest neighbor exchange of data in a ring topology.

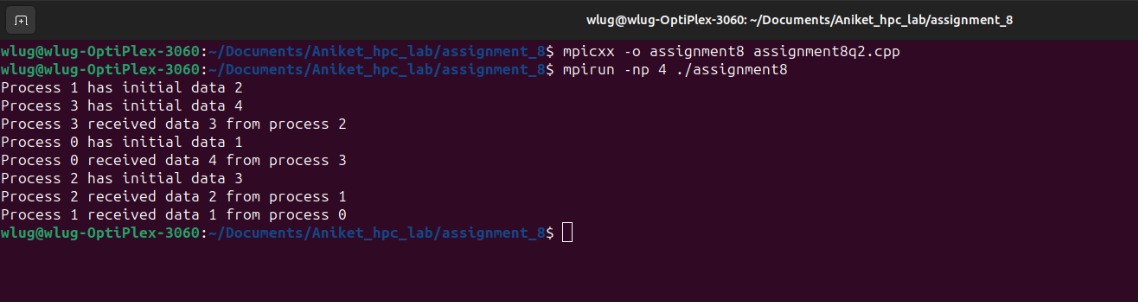
**Ans:**

**Blocking MPI Send & Receive for Nearest Neighbor Exchange in Ring Topology**

In a ring topology, each process exchanges data with its neighboring processes.

**Screenshots:**





**Explaination/Information:**

Each process sends data to its right neighbor and receives data from its left neighbor using MPI\_Sendrecv, which avoids deadlock and ensures the exchange is done in both directions.

**Q3.** Write a MPI program to find the sum of all the elements of an array A of size n. Elements of an array can be divided into two equals groups. The first [n/2] elements are added by the first process, P0, and last [n/2] elements the by second process, P1. The two sums then are added to get the final result.

**Ans:**

**Sum of Array Elements Using MPI**

In this program, the array is divided into two halves. Process P0 sums the first half, and process P1 sums the second half. The two sums are then added to get the final result.

**Screenshots:**

#include <mpi.h>

#include <iostream>

#include <vector>

using namespace std;

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

    int rank, size;

    const int n = 8;

    vector<int> A(n);

    int local\_sum = 0;

    MPI\_Init(&argc, &argv);

    MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

    MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

    if (rank == 0) {

        for (int i = 0; i < n; ++i) {

            A[i] = i + 1;

        }

    }

    MPI\_Bcast(A.data(), n, MPI\_INT, 0, MPI\_COMM\_WORLD);

    if (rank == 0) {

        for (int i = 0; i < n / 2; ++i) {

            local\_sum += A[i];

        }

    } else if (rank == 1) {

        for (int i = n / 2; i < n; ++i) {

            local\_sum += A[i];

        }

    }

    cout << "Process " << rank << " local sum: " << local\_sum << endl;

    int total\_sum;

    MPI\_Reduce(&local\_sum, &total\_sum, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

    if (rank == 0) {

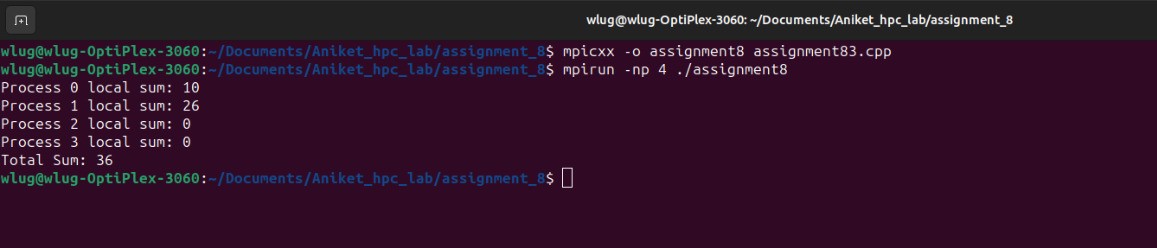
        cout << "Total Sum: " << total\_sum << endl;

    }

    MPI\_Finalize();

    return 0;

}



**Explaination/Information:**

* Process P0 and P1 each calculate the sum of half the array.
* MPI\_Reduce is used to gather the partial sums and compute the final result, which is printed by P0.

**Github Link:**

<https://github.com/AniketGhotkar/HPC_LAB_NEW/tree/main/practical%208>