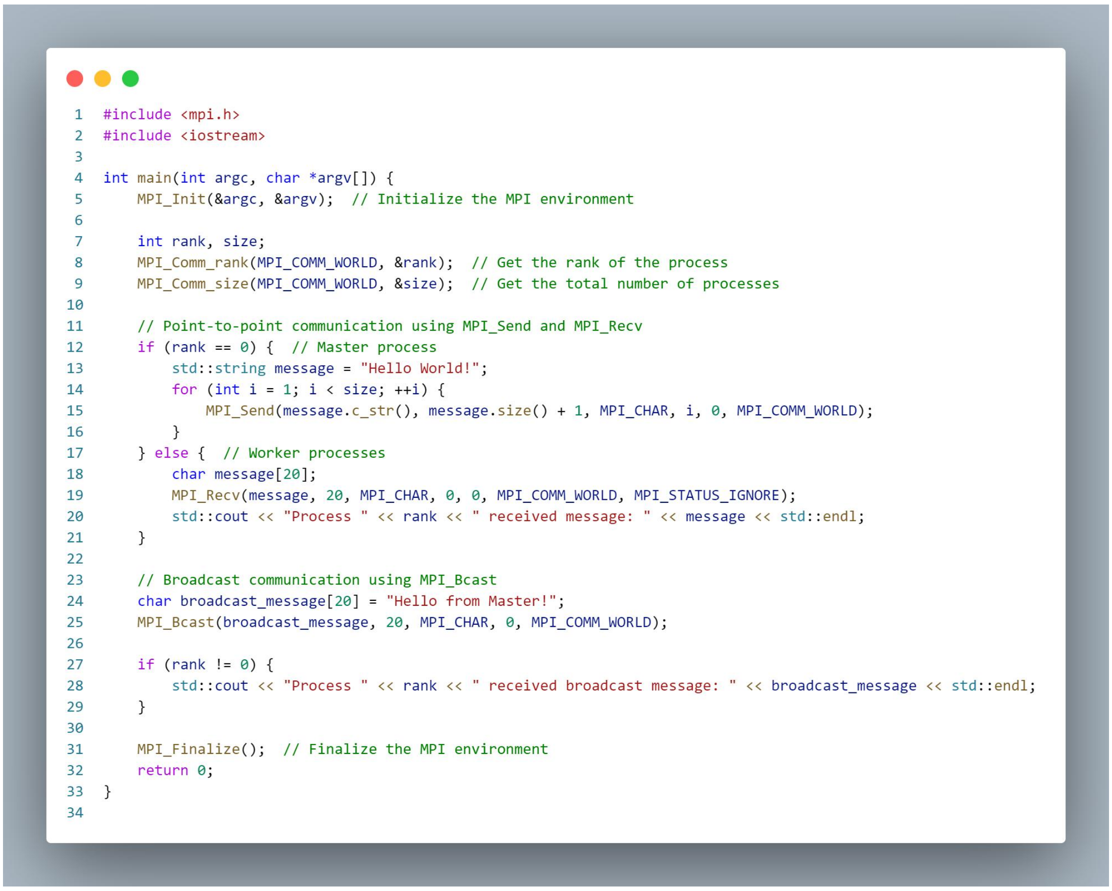
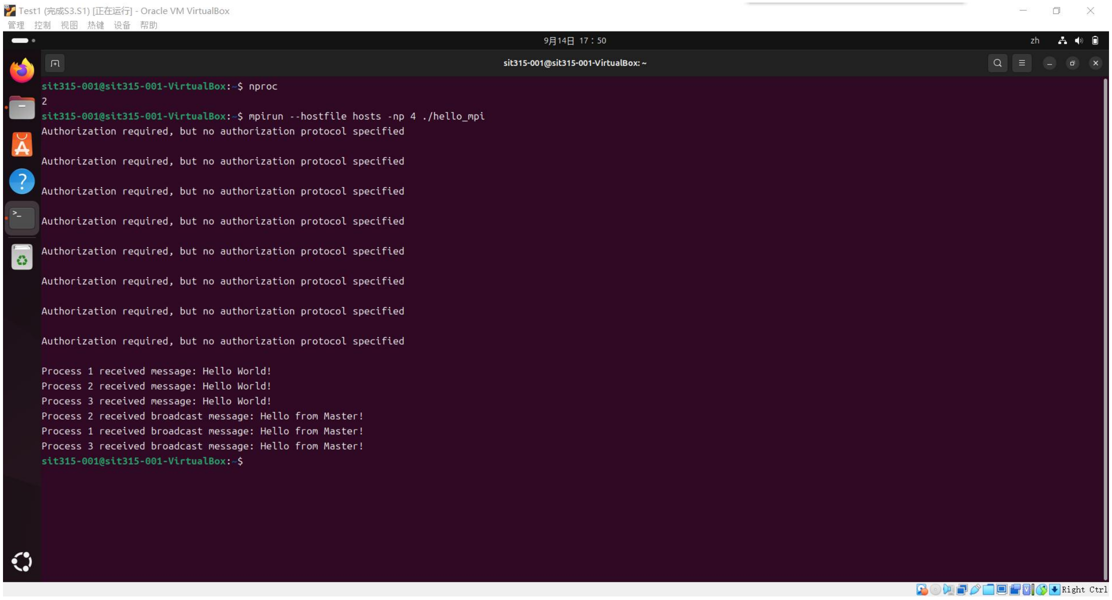
Activity 1

Code Implementation



```cpp
#include <mpi.h>
#include <iostream>

int main(int argc, char *argv[]) {
    MPI_Init(&argc, &argv);  // Initialize the MPI environment

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

    // Point-to-point communication using MPI_Send and MPI_Recv
    if (rank == 0) {  // Master process
        std::string message = "Hello World!";
        for (int i = 1; i < size; ++i) {
            MPI_Send(message.c_str(), message.size() + 1, MPI_CHAR, i, 0, MPI_COMM_WORLD);
        }
    } else {  // Worker processes
        char message[20];
        MPI_Recv(message, 20, MPI_CHAR, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        std::cout << "Process " << rank << " received message: " << message << std::endl;
    }

    // Broadcast communication using MPI_Bcast
    char broadcast_message[20] = "Hello from Master!";
    MPI_Bcast(broadcast_message, 20, MPI_CHAR, 0, MPI_COMM_WORLD);

    if (rank != 0) {
        std::cout << "Process " << rank << " received broadcast message: " << broadcast_message << std::endl;
    }

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

Results



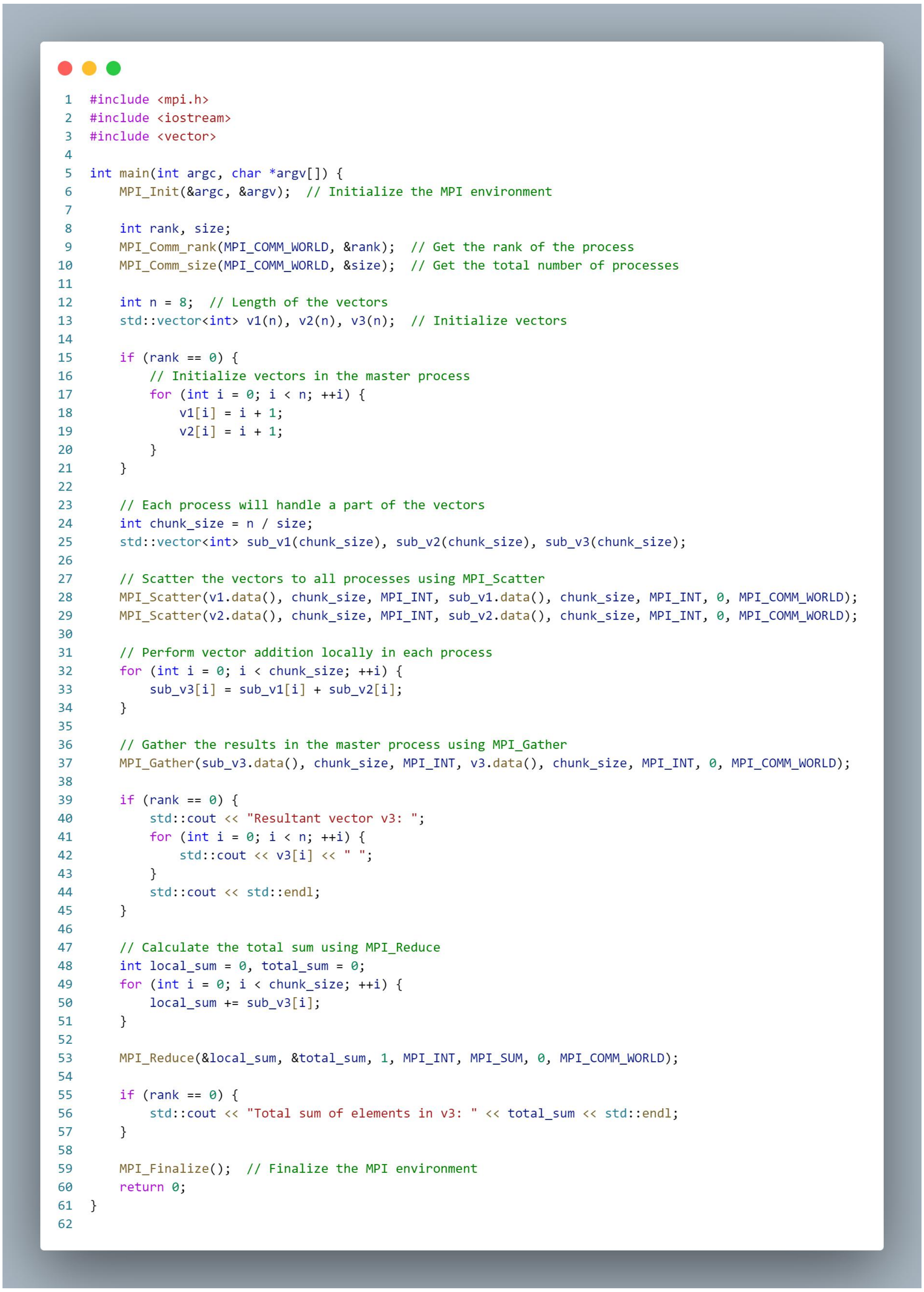
Test1 (完成S3.S1) [正在运行] - Oracle VM VirtualBox
管理 控制 视图 热键 设备 帮助
9月14日 17：50
sit315-001@sit315-001-VirtualBox: ~
Right Ctrl

```
sit315-001@sit315-001-VirtualBox:~$ nproc
2
sit315-001@sit315-001-VirtualBox:~$ mpirun --hostfile hosts -np 4 ./hello_mpi
Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Process 1 received message: Hello World!
Process 2 received message: Hello World!
Process 3 received message: Hello World!
Process 2 received broadcast message: Hello from Master!
Process 1 received broadcast message: Hello from Master!
Process 3 received broadcast message: Hello from Master!
sit315-001@sit315-001-VirtualBox:~$
```

In this activity, I successfully implemented both point-to-point and broadcast communication using MPI. All processes correctly received and printed the messages sent by the master process.

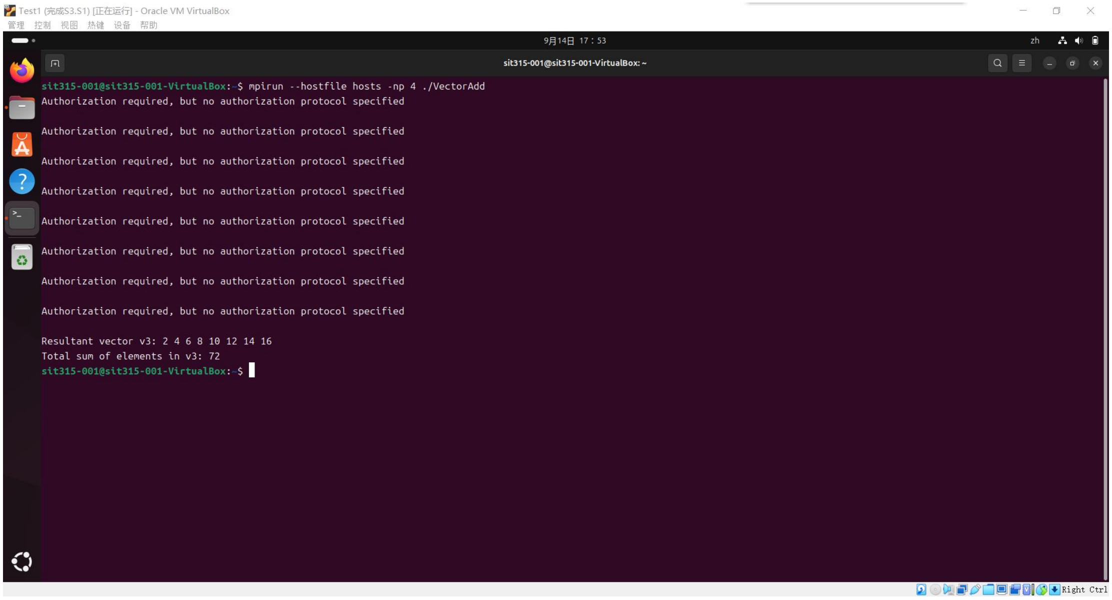
Activity 2

Code Implementation



```cpp
#include <mpi.h>
#include <iostream>
#include <vector>

int main(int argc, char *argv[]) {
    MPI_Init(&argc, &argv);  // Initialize the MPI environment

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

    int n = 8;  // Length of the vectors
    std::vector<int> v1(n), v2(n), v3(n);  // Initialize vectors

    if (rank == 0) {
        // Initialize vectors in the master process
        for (int i = 0; i < n; ++i) {
            v1[i] = i + 1;
            v2[i] = i + 1;
        }
    }

    // Each process will handle a part of the vectors
    int chunk_size = n / size;
    std::vector<int> sub_v1(chunk_size), sub_v2(chunk_size), sub_v3(chunk_size);

    // Scatter the vectors to all processes using MPI_Scatter
    MPI_Scatter(v1.data(), chunk_size, MPI_INT, sub_v1.data(), chunk_size, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Scatter(v2.data(), chunk_size, MPI_INT, sub_v2.data(), chunk_size, MPI_INT, 0, MPI_COMM_WORLD);

    // Perform vector addition locally in each process
    for (int i = 0; i < chunk_size; ++i) {
        sub_v3[i] = sub_v1[i] + sub_v2[i];
    }

    // Gather the results in the master process using MPI_Gather
    MPI_Gather(sub_v3.data(), chunk_size, MPI_INT, v3.data(), chunk_size, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        std::cout << "Resultant vector v3: ";
        for (int i = 0; i < n; ++i) {
            std::cout << v3[i] << " ";
        }
        std::cout << std::endl;
    }

    // Calculate the total sum using MPI_Reduce
    int local_sum = 0, total_sum = 0;
    for (int i = 0; i < chunk_size; ++i) {
        local_sum += sub_v3[i];
    }

    MPI_Reduce(&local_sum, &total_sum, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        std::cout << "Total sum of elements in v3: " << total_sum << std::endl;
    }

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

Results


Test1 (完成S3.S1) [正在运行] - Oracle VM VirtualBox
管理 控制 视图 热键 设备 帮助
9月14日 17：53
zh
sit315-001@sit315-001-VirtualBox: ~
Right Ctrl

```
sit315-001@sit315-001-VirtualBox:~$ mpirun --hostfile hosts -np 4 ./VectorAdd
Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Authorization required, but no authorization protocol specified

Resultant vector v3: 2 4 6 8 10 12 14 16
Total sum of elements in v3: 72
sit315-001@sit315-001-VirtualBox:~$ 
```

In this activity, I successfully implemented a distributed vector addition program using MPI. The program correctly distributed and computed the vector addition and calculated the total sum of the result vector.