CSE4001 - Parallel and Distributed Computing

Lab 21+22

Lab Assignment- 6

Submitted by: Alokam Nikhitha

Reg No:19BCE2555

QUESTION 1:

Write a C program to handle message passing in the MPI application interface, which allows processes to communicate with one another. Create two processes that will pass the number 20 from one to the other.

CODE:

```
#include <stdio.h>

#include <mpi.h>

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

int process_Rank, size_Of_Cluster, message_Item;

MPI_Init(&argc, &argv);

MPI_Comm_size(MPI_COMM_WORLD, &size_Of_Cluster);

MPI_Comm_rank(MPI_COMM_WORLD, &process_Rank);

if(process_Rank == 0){

message_Item = 20;

MPI_Send(&message_Item, 1, MPI_INT, 1, 1, MPI_COMM_WORLD);

printf("Message Sent from Process 0: %d\n", message_Item);

}

else if(process_Rank == 1){

MPI_Recv(&message_Item, 1, MPI_INT, 0, 1, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);

printf("Message Received in Process 1: %d\n", message_Item);

}

MPI_Finalize();

return 0;

}
```

CODE SNIPPETS:

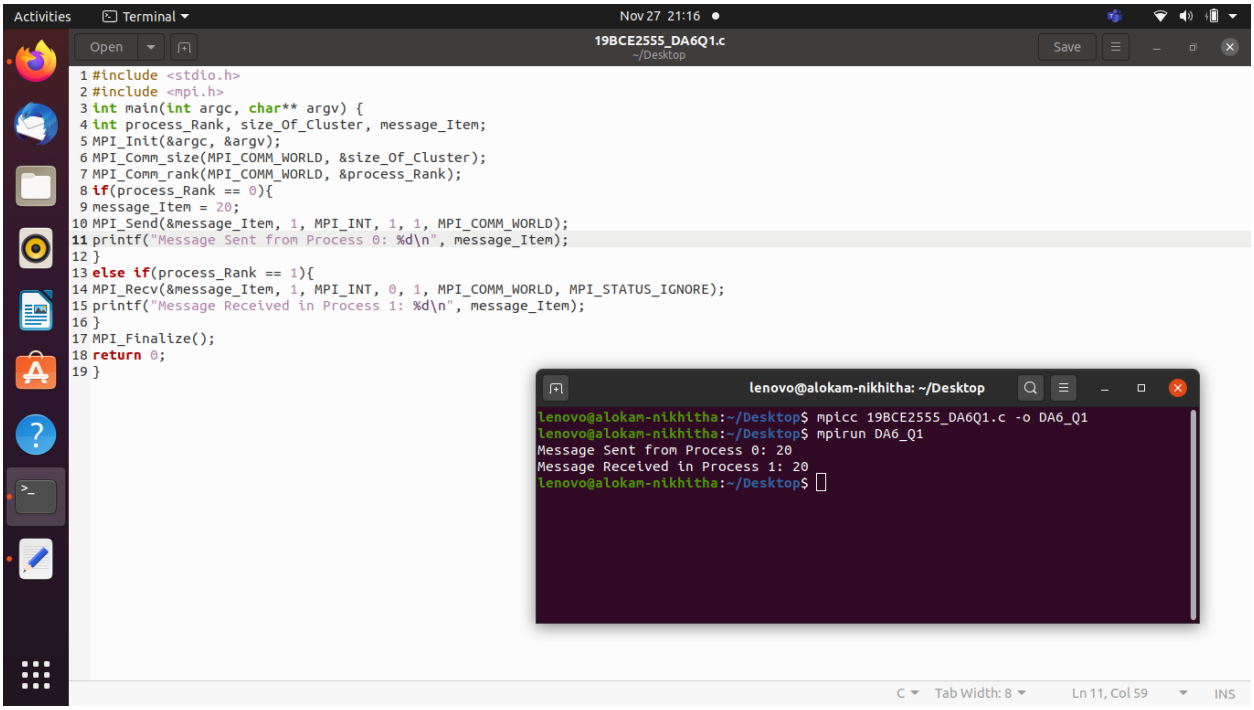
19BCE2555_DA6Q1.c

```c
#include <stdio.h>
#include <mpi.h>
int main(int argc, char** argv) {
int process_Rank, size_Of_Cluster, message_Item;
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &size_Of_Cluster);
MPI_Comm_rank(MPI_COMM_WORLD, &process_Rank);
if(process_Rank == 0){
message_Item = 20;
MPI_Send(&message_Item, 1, MPI_INT, 1, 1, MPI_COMM_WORLD);
printf("Message Sent from Process 0: %d\n", message_Item);
}
else if(process_Rank == 1){
MPI_Recv(&message_Item, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
printf("Message Received in Process 1: %d\n", message_Item);
}
MPI_Finalize();
return 0;
}
```

OUTPUT:

```
lenovo@alokam-nikhitha:~/Desktop$ mpicc 19BCE2555_DA6Q1.c -o DA6_Q1
lenovo@alokam-nikhitha:~/Desktop$ mpirun DA6_Q1
Message Sent from Process 0: 20
Message Received in Process 1: 20
lenovo@alokam-nikhitha:~/Desktop$
```

OUTPUT WITH CODE:



Result and Inferences:

- **Here we are passing the message between 2 Processes**
- **We can see that the Number 20 is passed from one Process to other**

QUESTION 2:

Write a C program to handle message passing in the MPI application interface using Group Operators: Scatter and Gather.

CODE:

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv) {
   int size, rank;
   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   int *globaldata=NULL;
   int localdata;
   if (rank == 0) {
      globaldata = malloc(size * sizeof(int) );
      for (int i=0; i<size; i++)
         globaldata[i] = 3*i+2;
      printf("Processor %d has data: ", rank);
      for (int i=0; i<size; i++)
         printf("%d ", globaldata[i]);
      printf("\n");
   }
   MPI_Scatter(globaldata, 1, MPI_INT, &localdata, 1, MPI_INT, 0,
MPI_COMM_WORLD);
```

```
    printf("Processor %d has data %d\n", rank, localdata);
    localdata *= 2;
    printf("Processor %d doubling the data, now has %d\n", rank, localdata);

    MPI_Gather(&localdata, 1, MPI_INT, globaldata, 1, MPI_INT, 0,
MPI_COMM_WORLD);
    if (rank == 0) {
        printf("Processor %d has data: ", rank);
        for (int i=0; i<size; i++)
            printf("%d ", globaldata[i]);
        printf("\n");
    }
    if (rank == 0)
        free(globaldata);
    MPI_Finalize();
    return 0;
}
```

Code Snippets:

Activities Text Editor Nov 27 21:17

Open 19BCE2555_DA6Q2.c ~/Desktop Save

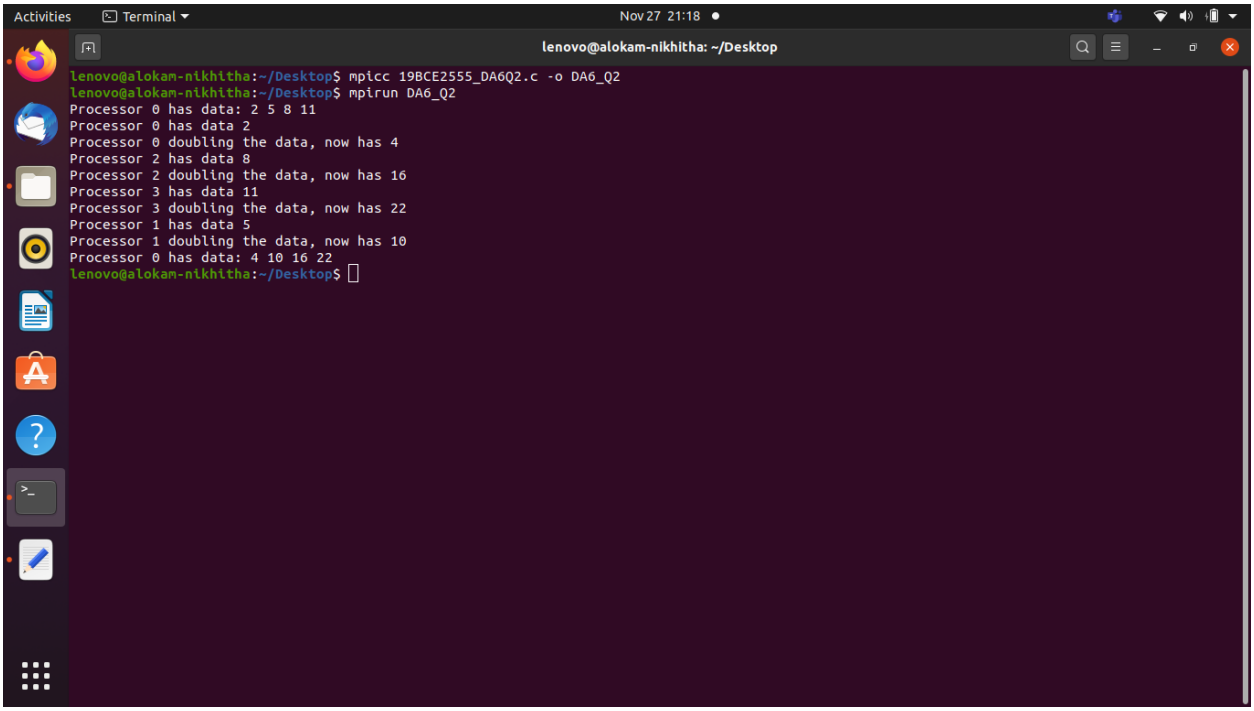
```c
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {
    int size, rank;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    int *globaldata=NULL;
    int localdata;

    if (rank == 0) {
        globaldata = malloc(size * sizeof(int) );
        for (int i=0; i<size; i++)
            globaldata[i] = 3*i+2;

        printf("Processor %d has data: ", rank);
        for (int i=0; i<size; i++)
            printf("%d ", globaldata[i]);
        printf("\n");
    }

    MPI_Scatter(globaldata, 1, MPI_INT, &localdata, 1, MPI_INT, 0, MPI_COMM_WORLD);

    printf("Processor %d has data %d\n", rank, localdata);
    localdata *= 2;
    printf("Processor %d doubling the data, now has %d\n", rank, localdata);

    MPI_Gather(&localdata, 1, MPI_INT, globaldata, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Processor %d has data: ", rank);
        for (int i=0; i<size; i++)
            printf("%d ", globaldata[i]);
        printf("\n");
```

C Tab Width: 8 Ln 1, Col 1 INS

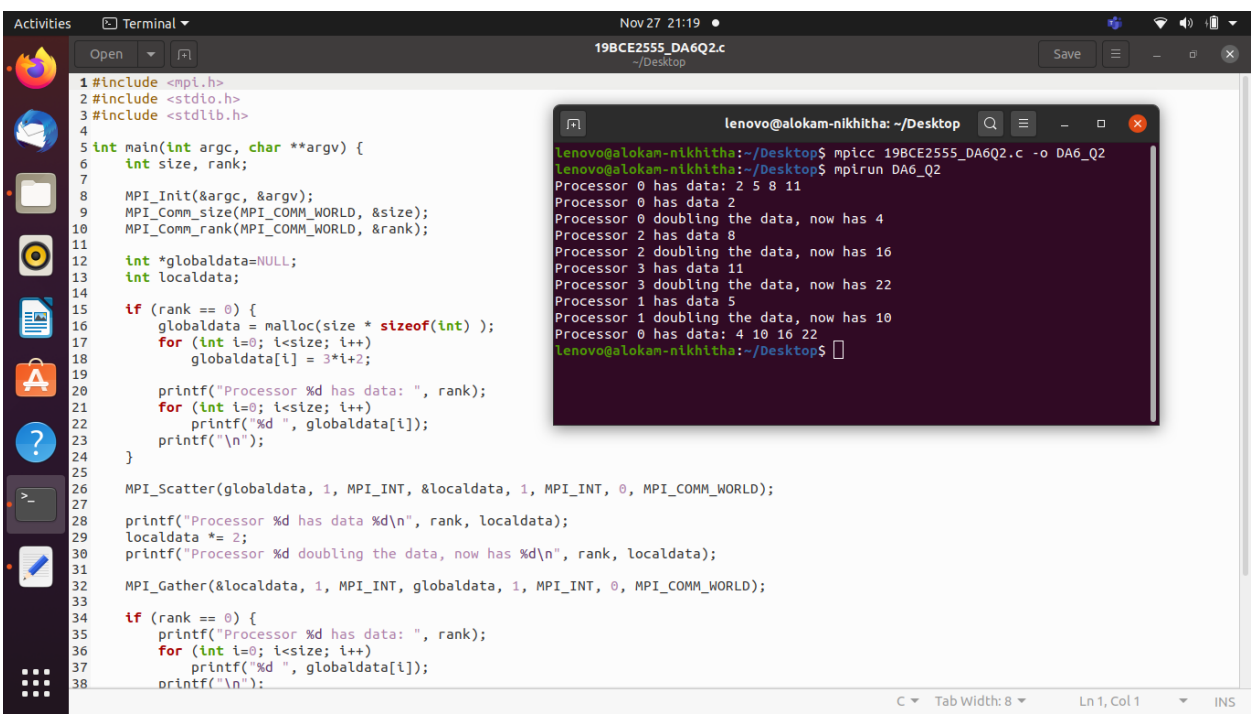
```c
        printf("\n");
    }

    if (rank == 0)
        free(globaldata);

    MPI_Finalize();
    return 0;
}
```

C Tab Width: 8 Ln 1, Col 1 INS

OUTPUT:



OUTPUT WITH CODE:



Result and Inferences:

- **We are passing the messages using MPI_Scatter and MPI_Gather Commands.**
- **Initialized the value in array as 3i+2 where i is the index of the array.**
- **We doubled its value during using different allocation named ldata.**
- **We used MPI_Gather command to read data of ldata allocation.**
- **Finally, We print the values of our initial allocation to check for the results.**