

CSE4001 - Parallel and Distributed Computing

Lab 21+22

Digital Assignment- 5

Submitted by: Alokam Nikhitha

Reg No:19BCE2555

QUESTION:

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

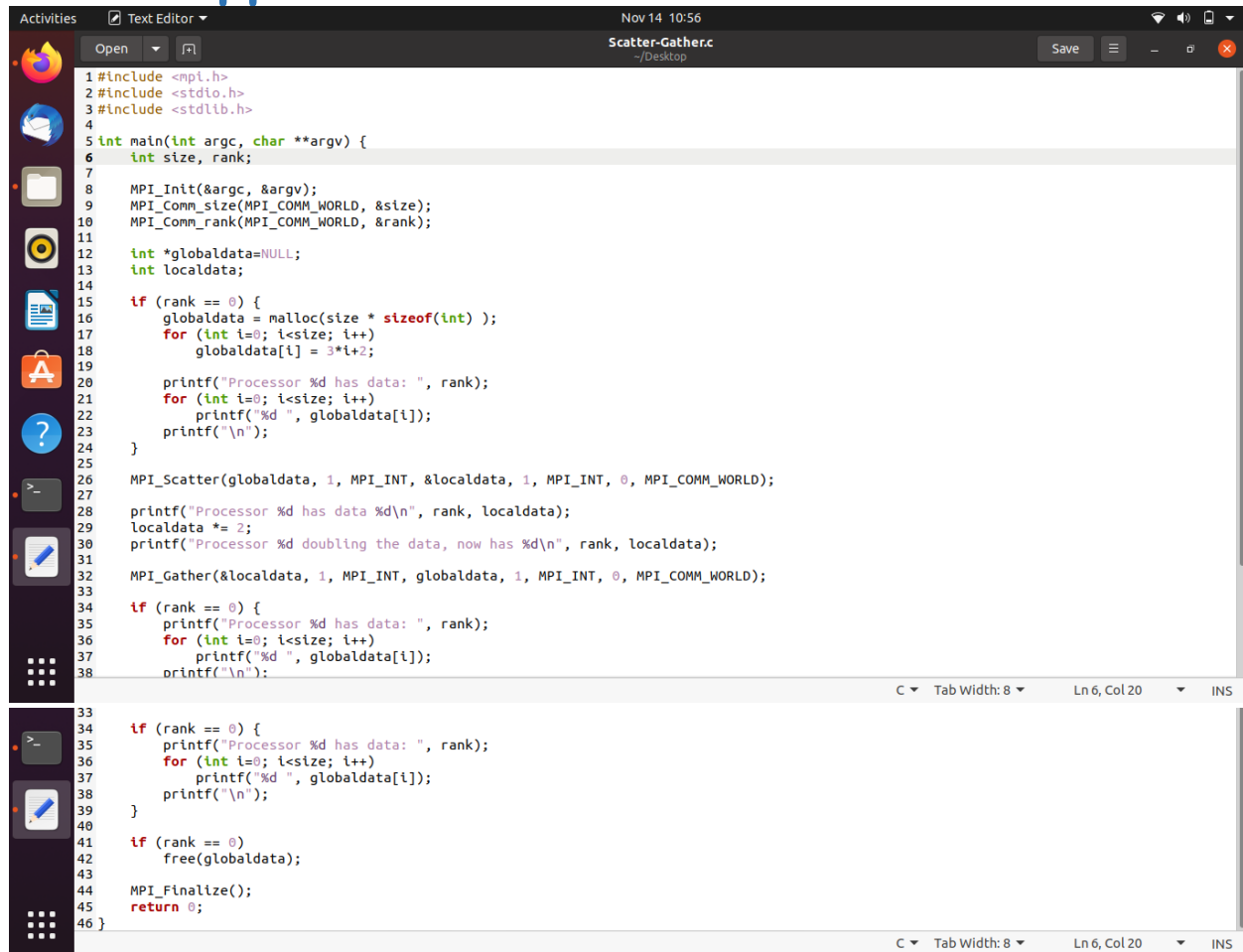
CODE:

```
#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;
}
```

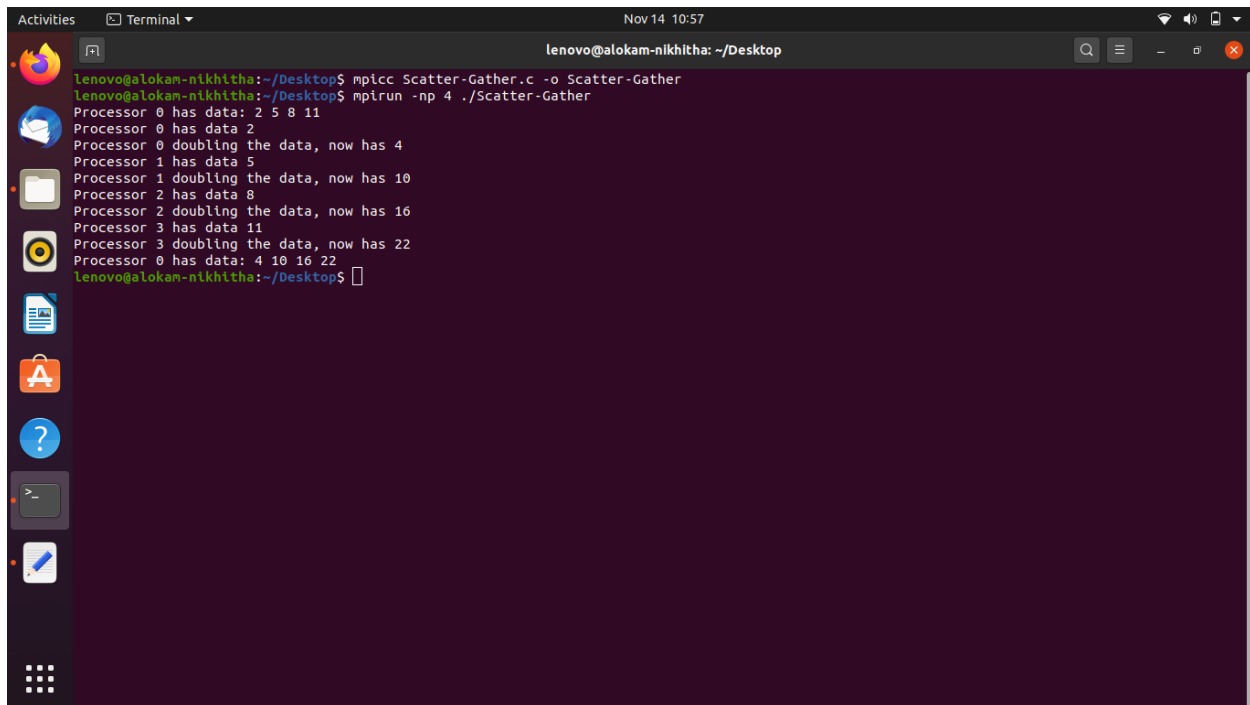
Code Snippets:



The screenshot shows a Linux desktop environment with a dark theme. On the left is a vertical dock with icons for Activities, Firefox, a terminal, a file manager, a music player, a document viewer, a terminal, a question mark, and a grid of application icons. The main window is a text editor titled "Scatter-Gather.c" located at "~/Desktop". The editor contains C code for an MPI program. The code is split across two tabs, each showing a different view of the same file. The top tab shows lines 1 through 38, and the bottom tab shows lines 33 through 46. The code implements a scatter-gather operation where each processor calculates data, then all processors gather the data, and finally, the root processor (rank 0) prints the global data.

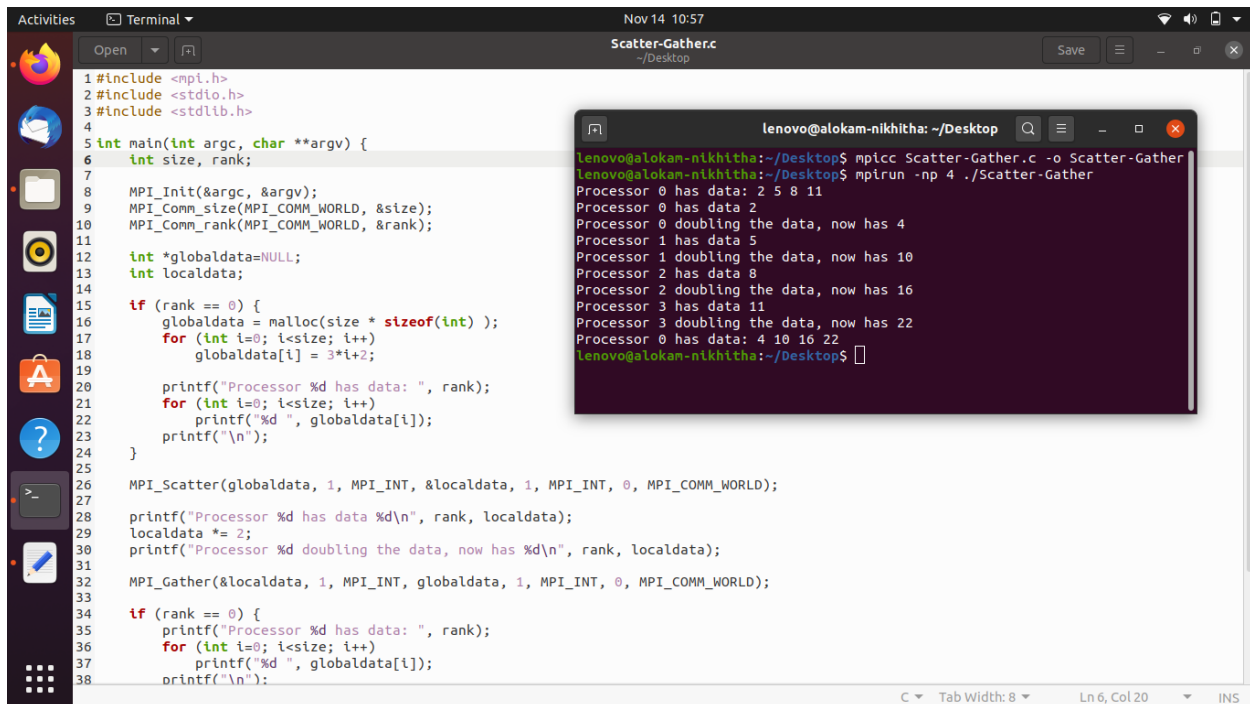
```
1 #include <mpi.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 int main(int argc, char **argv) {
6     int size, rank;
7
8     MPI_Init(&argc, &argv);
9     MPI_Comm_size(MPI_COMM_WORLD, &size);
10    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
11
12    int *globaldata=NULL;
13    int localdata;
14
15    if (rank == 0) {
16        globaldata = malloc(size * sizeof(int) );
17        for (int i=0; i<size; i++)
18            globaldata[i] = 3*i+2;
19
20        printf("Processor %d has data: ", rank);
21        for (int i=0; i<size; i++)
22            printf("%d ", globaldata[i]);
23        printf("\n");
24    }
25
26    MPI_Scatter(globaldata, 1, MPI_INT, &localdata, 1, MPI_INT, 0, MPI_COMM_WORLD);
27
28    printf("Processor %d has data %d\n", rank, localdata);
29    localdata *= 2;
30    printf("Processor %d doubling the data, now has %d\n", rank, localdata);
31
32    MPI_Gather(&localdata, 1, MPI_INT, globaldata, 1, MPI_INT, 0, MPI_COMM_WORLD);
33
34    if (rank == 0) {
35        printf("Processor %d has data: ", rank);
36        for (int i=0; i<size; i++)
37            printf("%d ", globaldata[i]);
38        printf("\n");
39    }
40
41    if (rank == 0)
42        free(globaldata);
43
44    MPI_Finalize();
45    return 0;
46 }
```

OUTPUT:



```
lenovo@alokam-nikhitha: ~/Desktop
lenovo@alokam-nikhitha:~/Desktop$ mpicc Scatter-Gather.c -o Scatter-Gather
lenovo@alokam-nikhitha:~/Desktop$ mpirun -np 4 ./Scatter-Gather
Processor 0 has data: 2 5 8 11
Processor 0 has data 2
Processor 0 doubling the data, now has 4
Processor 1 has data 5
Processor 1 doubling the data, now has 10
Processor 2 has data 8
Processor 2 doubling the data, now has 16
Processor 3 has data 11
Processor 3 doubling the data, now has 22
Processor 0 has data: 4 10 16 22
lenovo@alokam-nikhitha:~/Desktop$
```

OUTPUT WITH CODE:



```
1 #include <mpi.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 int main(int argc, char **argv) {
6     int size, rank;
7
8     MPI_Init(&argc, &argv);
9     MPI_Comm_size(MPI_COMM_WORLD, &size);
10    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
11
12    int *globaldata=NULL;
13    int localdata;
14
15    if (rank == 0) {
16        globaldata = malloc(size * sizeof(int));
17        for (int i=0; i<size; i++)
18            globaldata[i] = 3*i+2;
19
20        printf("Processor %d has data: ", rank);
21        for (int i=0; i<size; i++)
22            printf("%d ", globaldata[i]);
23        printf("\n");
24    }
25
26    MPI_Scatter(globaldata, 1, MPI_INT, &localdata, 1, MPI_INT, 0, MPI_COMM_WORLD);
27
28    printf("Processor %d has data %d\n", rank, localdata);
29    localdata *= 2;
30    printf("Processor %d doubling the data, now has %d\n", rank, localdata);
31
32    MPI_Gather(&localdata, 1, MPI_INT, globaldata, 1, MPI_INT, 0, MPI_COMM_WORLD);
33
34    if (rank == 0) {
35        printf("Processor %d has data: ", rank);
36        for (int i=0; i<size; i++)
37            printf("%d ", globaldata[i]);
38        printf("\n");
39    }
```