

# CSE 4001

## PARALLEL AND DISTRIBUTED COMPUTING



### Lab Assessment – 3

L27+L28 | PLBG04

Dr. Narayanan Prasanth

FALL SEMESTER 2021-22

by

**SHARADINDU ADHIKARI**

19BCE2105

**1. MPI\_Bsend is the asynchronous blocking send (with user provided Buffering), it will block until a copy of the buffer is passed. Write a MPI program to establish a point to point communication between two process using the above function. (3)**

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
int main(int argc, char **argv)
{
    MPI_Init(NULL, NULL);
    int size;
    int my_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank); // get current process id
    switch (my_rank)
    {
        case 0:
        {
            int buffer_attached_size = MPI_BSEND_OVERHEAD + sizeof(int); //
            size of buffer attached to message
            char buffer_attached = (char)malloc(buffer_attached_size);
            MPI_Buffer_attach(buffer_attached, buffer_attached_size);

            printf("Sharadindu Adhikari\n19BCE2105\n");

            int message = 12345;
            printf("Process %d sent value : %d.\n", my_rank, message); //
            process 0 sends 12345
            MPI_Bsend(&message, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
            MPI_Buffer_detach(&buffer_attached, &buffer_attached_size);
            free(buffer_attached);
            break;
        }

        case 1:
        {
            int received;
            MPI_Recv(&received, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
            MPI_STATUS_IGNORE); // process 1 receives 12345
```

```

printf("Process %d received value : %d.\n", my_rank, received);
break;
}
}

MPI_Finalize();
}

```

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <mpi.h>
4 int main(int argc, char **argv)
5 {
6     MPI_Init(NULL, NULL);
7     int size;
8     int my_rank;
9     MPI_Comm_rank(MPI_COMM_WORLD, &my_rank); // get current process id
10    switch (my_rank)
11    {
12
13    case 0:
14    {
15        int buffer_attached_size = MPI_BSEND_OVERHEAD + sizeof(int); // size of buffer attached to message
16        char buffer_attached = (char)malloc(buffer_attached_size);
17        MPI_Buffer_attach(buffer_attached, buffer_attached_size);
18
19        printf("Sharadindu Adhikari\n19BCE2105\n");
20
21        int message = 12345;
22        printf("Process %d sent value : %d.\n", my_rank, message); // process 0 sends 12345
23        MPI_Bsend(&message, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
24        MPI_Buffer_detach(&buffer_attached, &buffer_attached_size);
25        free(buffer_attached);
26        break;
27    }
28    }
29 }

```

## OUTPUT:

```

shara-d@Rohans-Workstation:~/PDC$ cd Lab3
shara-d@Rohans-Workstation:~/PDC/Lab3$ mpicc prob1.c -o prob1
prob1.c: In function 'main':
prob1.c:16:32: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
16 |         char buffer_attached = (char)malloc(buffer_attached_size);
    |                                ^
prob1.c:17:27: warning: passing argument 1 of 'MPI_Buffer_attach' makes pointer from integer without a cast [-Wint-conversion]
17 |         MPI_Buffer_attach(buffer_attached, buffer_attached_size);
    |                           ^~~~~~
    |                           |
    |                           char
In file included from prob1.c:3:
/usr/lib/x86_64-linux-gnu/openmpi/include/mpi.h:1335:44: note: expected 'void *' but argument is of type 'char'
1335 |     OMPI_DECLSPEC int MPI_Buffer_attach(void *buffer, int size);
    |                                     ~~~~~^~~~~
prob1.c:25:14: warning: passing argument 1 of 'free' makes pointer from integer without a cast [-Wint-conversion]
25 |         free(buffer_attached);
    |         ~~~~~^~~~~
    |         |
    |         char
In file included from prob1.c:2:
/usr/include/stdlib.h:565:25: note: expected 'void *' but argument is of type 'char'
565 |     extern void free (void *__ptr) __THROW;
    |                     ~~~~~^~~~~
shara-d@Rohans-Workstation:~/PDC/Lab3$ mpirun -np 4 ./prob1

WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

Local host: Rohans-Workstation
-----
[Rohans-Workstation:01062] 1 more process has sent help message help-btl-vader.txt / cma-permission-denied
[Rohans-Workstation:01062] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
Sharadindu Adhikari
19BCE2105
Process 0 sent value : 12345.
Process 1 received value : 12345.
[Rohans-Workstation:01071] *** Process received signal ***
[Rohans-Workstation:01071] Signal: Segmentation fault (11)
[Rohans-Workstation:01071] Signal code: Address not mapped (1)
[Rohans-Workstation:01071] Failing at address: 0x58
[Rohans-Workstation:01071] [ 0] /lib/x86_64-linux-gnu/libc.so.6(0x46210)[0x7f6365446210]
[Rohans-Workstation:01071] [ 1] /lib/x86_64-linux-gnu/libc.so.6(cfree+0x20)[0x7f636549d870]
[Rohans-Workstation:01071] [ 2] ./prob1(+0x13a7)[0x7f63657653a7]

```

**2. Develop a MPI program to perform point to point communication between two processes using the following send functions (Standard, Ready, Synchronous) in blocking mode. Compute the time taken by each send functions and find the optimal one. (3)**

**Standard send:**

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

int main(int argc, char *argv[])
{
    MPI_Init(&argc, &argv);

    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int my_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    if (my_rank == 0)
    {
        printf("\n19BCE2105, Sharadindu Adhikari\n\n");

        int buffer_sent = 12345;
        printf("[Standard] MPI process %d sends value %d to process 1.\n",
my_rank, buffer_sent);
        double start = MPI_Wtime();
        MPI_Send(&buffer_sent, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
        double end = MPI_Wtime();
        printf("[Standard] MPI process %d send value %d in %f seconds.\n",
my_rank, buffer_sent, end - start);
    }
    else if (my_rank == 1)
    {
        int received;
        printf("[Standard] MPI process %d receiving from process 0.\n",
my_rank);
        double start = MPI_Wtime();
```

```

    MPI_Recv(&received, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
    double end = MPI_Wtime();
    printf("[Standard] MPI process %d received value %d in %f
seconds.\n", my_rank, received, end - start);
}

MPI_Finalize();
return 0;
}

```

```

2_std.c > main(int, char *[])
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <mpi.h>
4
5  int main(int argc, char *argv[])
6  {
7      MPI_Init(&argc, &argv);
8
9      int size;
10     MPI_Comm_size(MPI_COMM_WORLD, &size);
11
12     int my_rank;
13     MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
14
15     if (my_rank == 0)
16     {
17         printf("\n19BCE2105, Sharadindu Adhikari\n\n");
18
19         int buffer_sent = 12345;
20         printf("[Standard] MPI process %d sends value %d to process 1.\n", my_rank, buffer_sent);
21         double start = MPI_Wtime();
22         MPI_Send(&buffer_sent, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
23         double end = MPI_Wtime();
24         printf("[Standard] MPI process %d send value %d in %f seconds.\n", my_rank, buffer_sent, end - start);
25     }
26     else if (my_rank == 1)
27     {
28         int received;
29         printf("[Standard] MPI process %d receiving from process 0.\n", my_rank);
30         double start = MPI_Wtime();
31         MPI_Recv(&received, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
32         double end = MPI_Wtime();
33         printf("[Standard] MPI process %d received value %d in %f seconds.\n", my_rank, received, end - start);

```

```

shara-d@Rohans-Workstation:~/PDC$ cd Lab3
shara-d@Rohans-Workstation:~/PDC/Lab3$ mpicc 2_std.c -o 2_std
shara-d@Rohans-Workstation:~/PDC/Lab3$ mpirun -np 2 ./2_std

WARNING: Linux kernel CMA support was requested via the
bt1_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

Local host: Rohans-Workstation

19BCE105, Sharadindu Adhikari

Task 1: Received 7 char(s) from task 0 with tag 1
Task 1: Received 7 char(s) from task 0 with tag 1
Task 0: buffered send (1) buffered , process will block on detach. Time: 0.00
Task 0: buffered send (2), process may will block here. Time: 0.00
[Rohans-Workstation:02211] 1 more process has sent help message help-btl-vader.txt / cma-permission-denied
[Rohans-Workstation:02211] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
shara-d@Rohans-Workstation:~/PDC/Lab3$

```

**Ready send:**

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
int main(int argc, char* argv[])
{

MPI_Init(&argc, &argv);
int size;
MPI_Comm_size(MPI_COMM_WORLD, &size);
enum role_ranks { SENDER, RECEIVER };
int my_rank;
double t1, t2;
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
switch(my_rank)
{
case SENDER:
{

printf("\n19BCE2105, Sharadindu Adhikari\n\n");

MPI_Barrier(MPI_COMM_WORLD);
int send=1908;
printf("MPI process %d sends value %d.\n\n", my_rank, send);
MPI_Rsend(&send, 1, MPI_INT, RECEIVER, 0, MPI_COMM_WORLD);
break;
}
case RECEIVER:
{
int received;
MPI_Request request;
MPI_Irecv(&received, 1, MPI_INT, SENDER, 0, MPI_COMM_WORLD,
&request);
MPI_Barrier(MPI_COMM_WORLD);
MPI_Wait(&request, MPI_STATUS_IGNORE);
printf("MPI process %d receives value %d.\n", my_rank, received);
break;
}
}
MPI_Finalize();
return 0;
}
```

Visual Studio Code interface showing a C program for MPI communication. The Explorer sidebar on the left lists files under "PDC [WSL: UBUNTU]", including "2\_ready.c" which is selected. The main editor displays the code for "2\_ready.c", which includes `<stdio.h>`, `<stdlib.h>`, and `<mpi.h>`, and defines a `main` function that initializes MPI, sets up a world, and performs a send-recv operation.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <mpi.h>
4 int main(int argc, char* argv[])
5 {
6
7     MPI_Init(&argc, &argv);
8     int size;
9     MPI_Comm_size(MPI_COMM_WORLD, &size);
10    enum role_ranks { SENDER, RECEIVER };
11    int my_rank;
12    double t1, t2;
13    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
14    switch(my_rank)
15    {
16
```

The bottom panel shows the TERMINAL output, which includes the command to compile and run the program, a warning about CMA support, and the MPI process output showing a successful exchange of the value 1908.

```
shara-d@Rohans-Workstation:~/PDC/Lab3$ mpicc 2_ready.c -o 2_ready
shara-d@Rohans-Workstation:~/PDC/Lab3$ mpirun -np 2 ./2_ready
WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

-----
Local host: Rohans-Workstation
-----
19BCE2105, Sharadindu Adhikari

MPI process 0 sends value 1908.

MPI process 1 receives value 1908.
[Rohans-Workstation:03873] 1 more process has sent help message help-btl-vader.txt / cma-permission-denied
[Rohans-Workstation:03873] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
shara-d@Rohans-Workstation:~/PDC/Lab3$
```

### Synchronous send:

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
int main(int argc, char* argv[])
{
    MPI_Init(&argc, &argv);
    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    enum role_ranks { SENDER, RECEIVER };
    int my_rank;
    double t1, t2;
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    switch(my_rank)
    {
    case SENDER:
    {
        MPI_Barrier(MPI_COMM_WORLD);
        int Ssend=1908;
        printf("Ssend : MPI process %d sends value %d.\n\n", my_rank, Ssend);
        MPI_Ssend(&Ssend, 1, MPI_INT, RECEIVER, 0, MPI_COMM_WORLD);
    }
    }
}
```

```

break;
}
case RECEIVER:
{
printf("\n19BCE2105, Sharadindu Adhikari\n\n");
int received;
MPI_Request request;
MPI_Irecv(&received, 1, MPI_INT, SENDER, 0, MPI_COMM_WORLD,
&request);

MPI_Barrier(MPI_COMM_WORLD);
MPI_Wait(&request, MPI_STATUS_IGNORE);
printf("Ssend : MPI process %d receives value %d.\n", my_rank, received);
break;
}
}
MPI_Finalize();
return 0;
}

```

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** Shows the project structure under 'PDC [WSL: UBUNTU]'. Files include .vscode, In-Lab, Lab1, Lab2, Lab3, 1x, 1x.c, 2\_ready, 2\_ready.c, 2\_std, 2\_std copy.c, 2\_std.c, 2\_sync, 2\_sync.c, 2rsend, 2sn, 2sn.c, 2stend, 2stsend, 2ulti, 2ulti.c, 2xc, PDC\_03\_02a.cZone.Identifier, prob1, prob1.c, prob2ready, prob2standard, prob2synchronous, prob3, prob3.c, Lab4, and Midterm.
- Main Editor:** Displays the code for '2\_sync.c'. The code includes headers, initializes MPI, and implements the SENDER and RECEIVER cases. The SENDER case sends a value of 1908 to the RECEIVER process.
- Terminal:** Shows the compilation and execution of the program. The output includes:
 

```

shara-d@Rohans-Workstation:~/PDC/Lab3$ mpicc 2_sync.c -o 2_sync
shara-d@Rohans-Workstation:~/PDC/Lab3$ mpirun -np 2 ./2_sync

WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

Local host: Rohans-Workstation

19BCE2105, Sharadindu Adhikari

Ssend : MPI process 0 sends value 1908.

Ssend : MPI process 1 receives value 1908.
[Rohans-Workstation:03742] 1 more process has sent help message help-btl-vader.txt / cma-permission-denied
[Rohans-Workstation:03742] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
shara-d@Rohans-Workstation:~/PDC/Lab3$

```



**ALTERNATIVE SOLUTION TO PROBLEM 2:**

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(int argc, char **argv)
{
    printf("19BCE2105, Sharadindu Adhikari\n\n");
    // Initialize the MPI environment
    MPI_Init(NULL, NULL);
    // Find out rank, size
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    // We are assuming at least 2 processes for this task
    if (world_size < 2)
    {
        fprintf(stderr, "World size must be greater than 1 for %s\n",
argv[0]);
        MPI_Abort(MPI_COMM_WORLD, 1);
    }
    int number, i;
    double t1;
    t1 = MPI_Wtime();
    printf("-----\n");
    printf("Standard Blocking Mode\n");
    for (i = 0; i < 100; i++)
    {
        if (world_rank == 0)
        {
            // If we are rank 0, set the number to -1 and send it to process
1
            number = -1;
            MPI_Send(
                /* data = */ &number,
                /* count = */ 1,
                /* datatype = */ MPI_INT,
                /* destination = */ 1,
                /* tag = */ 0,
                /* communicator = */ MPI_COMM_WORLD);
        }
        else if (world_rank == 1)
        {

```

```

        MPI_Recv(
            /* data = */ &number,
            /* count = */ 1,
            /* datatype = */ MPI_INT,
            /* source = */ 0,
            /* tag = */ 0,
            /* communicator = */ MPI_COMM_WORLD,
            /* status = */ MPI_STATUS_IGNORE);
        //printf("Process 1 received number %d from process 0\n",
number);
    }
}
double t2;
t2 = MPI_Wtime();
printf("Time taken: %f\n", t2 - t1);
printf("-----\n");
double t3;
t3 = MPI_Wtime();
printf("Ready Blocking Mode\n");
for (i = 0; i < 100; i++)
{
    if (world_rank == 0)
    {
        // If we are rank 0, set the number to -1 and send it to process
1
        number = -1;
        MPI_Rsend(
            /* data = */ &number,
            /* count = */ 1,
            /* datatype = */ MPI_INT,
            /* destination = */ 1,
            /* tag = */ 0,
            /* communicator = */ MPI_COMM_WORLD);
    }
    else if (world_rank == 1)
    {
        MPI_Recv(
            /* data = */ &number,
            /* count = */ 1,
            /* datatype = */ MPI_INT,
            /* source = */ 0,
            /* tag = */ 0,
            /* communicator = */ MPI_COMM_WORLD,
            /* status = */ MPI_STATUS_IGNORE);
    }
}

```

```

        //printf("Process 1 received number %d from process 0\n",
number);
    }
}
double t4;
t4 = MPI_Wtime();
printf("Time taken: %lf\n", t4 - t3);
printf("-----\n");
double t5;
t5 = MPI_Wtime();
printf("Synchronous Blocking Mode\n");
for (i = 0; i < 100; i++)
{
    if (world_rank == 0)
    {
        // If we are rank 0, set the number to -1 and send it to process
1
        number = -1;
        MPI_Ssend(
            /* data = */ &number,
            /* count = */ 1,
            /* datatype = */ MPI_INT,
            /* destination = */ 1,
            /* tag = */ 0,
            /* communicator = */ MPI_COMM_WORLD);
    }
    else if (world_rank == 1)
    {
        MPI_Recv(
            /* data = */ &number,
            /* count = */ 1,
            /* datatype = */ MPI_INT,
            /* source = */ 0,
            /* tag = */ 0,
            /* communicator = */ MPI_COMM_WORLD,
            /* status = */ MPI_STATUS_IGNORE);
        //printf("Process 1 received number %d from process 0\n",
number);
    }
}
double t6;
t6 = MPI_Wtime();
printf("Time taken: %lf\n", t6 - t5);
printf("-----\n");

```

```
MPI_Finalize();
```

```
}
```

```

1 #include <mpi.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <time.h>
5 int main(int argc, char **argv)
6 {
7     printf("19BCE2105, Sharadindu Adhikari\n\n");
8     // Initialize the MPI environment
9     MPI_Init(NULL, NULL);
10    // Find out rank, size
11    int world_rank;
12    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
13    int world_size;
14    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
15    // We are assuming at least 2 processes for this task
16    if (world_size < 2)
17    {
18        fprintf(stderr, "World size must be greater than 1 for %s\n", argv[0]);
19        MPI_Abort(MPI_COMM_WORLD, 1);
20    }
21    int number, i;

```

Output:

```

shara-d@Rohans-Workstation:~/PDC$ cd Lab3
shara-d@Rohans-Workstation:~/PDC/Lab3$ mpicc 2ulti.c -o 2ulti
shara-d@Rohans-Workstation:~/PDC/Lab3$ mpirun -np 2 ./2ulti
19BCE2105, Sharadindu Adhikari

-----
WARNING: Linux kernel CMA support was requested via the
btl_vader single copy mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

-----
Local host: Rohans-Workstation
-----
Standard Blocking Mode
-----
Standard Blocking Mode
Time taken: 0.000378
-----
Ready Blocking Mode
Time taken: 0.000027
-----
Synchronous Blocking Mode
Time taken: 0.000486
-----
Ready Blocking Mode
Time taken: 0.000049
-----
Synchronous Blocking Mode
Time taken: 0.000297
-----
Time taken: 0.000181
-----
[Rohans-Workstation:04151] 1 more process has sent help message help-btl-vader.txt / cma-permission-denied
[Rohans-Workstation:04151] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
shara-d@Rohans-Workstation:~/PDC/Lab3$

```

**Inference:** From the output, we can definitely conclude that Ready send function is the most optimal one!

Q3:

**2. A communicator has 3 process (ID: 0, 1 and 2) such that process 0 sends a binary value 001 to process 1, in-turn process 1 increment the value by 1 (i.e.010) and send the same to process 2, in-turn process 2 increment the value by 1 (i.e.011) and send the same to process 0. Continue the process until the binary value becomes 111. Develop a MPI program to implement the above scenario.** (4)

```
#include <mpi.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char **argv)

{
    MPI_Init(NULL, NULL);
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    char binaryno[4];
    if (world_rank == 0)

    {
        printf("Sharadindu Adhikari\n19BCE2105\n");

        strcpy(binaryno, "001");
        MPI_Send(&binaryno, 4, MPI_CHAR, 1, 0, MPI_COMM_WORLD);
        printf("MessageSentfromProcess%d:%s\n", world_rank, binaryno);
        MPI_Recv(&binaryno, 4, MPI_CHAR, 2, 2, MPI_COMM_WORLD,
                MPI_STATUS_IGNORE);
        printf("FinalMessageReceivedatProcess%d:%s\n", world_rank,
binaryno);
    }

    else if (world_rank == 1)
    {
        MPI_Recv(&binaryno, 4, MPI_CHAR, 0, 0, MPI_COMM_WORLD,
                MPI_STATUS_IGNORE);
        printf("MessageReceivedatProcess%d:%s\n", world_rank, binaryno);
        strcpy(binaryno, "010");
        MPI_Send(&binaryno, 4, MPI_CHAR, 2, 1, MPI_COMM_WORLD);
        printf("MessageSentfromProcess%d:%s\n", world_rank, binaryno);
    }
}
```

```

else if (world_rank == 2)
{
    MPI_Recv(&binaryno, 4, MPI_CHAR, 1, 1, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
    printf("MessageReceivedatProcess%d:%s\n", world_rank, binaryno);
    strcpy(binaryno, "011");
    MPI_Send(&binaryno, 4, MPI_CHAR, 0, 2, MPI_COMM_WORLD);
    printf("MessageSentfromProcess%d:%s\n", world_rank, binaryno);
}

MPI_Finalize();
}

```

## OUTPUT:

```

Lab3 > C prob3.c > main(int, char **)
4  int main(int argc, char **argv)
5  {
6      MPI_Init(NULL, NULL);
7      int world_rank;
8      MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
9      char binaryno[4];
10     if (world_rank == 0)
11     {
12         printf("Sharadindu Adhikari\n19BCE2105\n");
13
14         strcpy(binaryno, "001");
15         MPI_Send(&binaryno, 4, MPI_CHAR, 1, 0, MPI_COMM_WORLD);
16         printf("MessageSentfromProcess%d:%s\n", world_rank, binaryno);
17         MPI_Recv(&binaryno, 4, MPI_CHAR, 2, 2, MPI_COMM_WORLD,
18                 MPI_STATUS_IGNORE);

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

shara-d@Rohans-Workstation:~/PDC/Lab3$ mpicc prob3.c -o prob3
shara-d@Rohans-Workstation:~/PDC/Lab3$ mpirun -np 4 ./prob3

WARNING: Linux kernel CMA support was requested via the
btl_vader_single_copy_mechanism MCA variable, but CMA support is
not available due to restrictive ptrace settings.

The vader shared memory BTL will fall back on another single-copy
mechanism if one is available. This may result in lower performance.

Local host: Rohans-Workstation

[Rohans-Workstation:01369] 2 more processes have sent help message help-btl-vader.txt / cma-permission-denied
[Rohans-Workstation:01369] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help / error messages
Sharadindu Adhikari
19BCE2105
MessageSentfromProcess0:001
FinalMessageReceivedatProcess0:011
MessageReceivedatProcess1:001
MessageSentfromProcess1:010
MessageReceivedatProcess2:010
MessageSentfromProcess2:011
[Rohans-Workstation:01369] 1 more process has sent help message help-btl-vader.txt / cma-permission-denied
shara-d@Rohans-Workstation:~/PDC/Lab3$

```

Ln 39, Col 2 Spaces: 4 UTF-8 LF C Linux