Fall Sem 2021-22

Assignment: 6

Date: 20/10/21

Name: Ishan Sagar Jogalekar

Reg no: 19BCE2250

Course: Parallel and distributed computing LAB - CSE4001

Slot: L55+L56

Aim: Consider a suitable instance that has MPI routines to assign different tasks to different processors.

Ans –

SOURCE CODE:

```
#include <stdio.h>
#include <mpi.h>
int main(int argc,char** argv){
    //Intializing MPI env
    MPI_Init(NULL,NULL);

   //Get no of process
   int no_process;
   MPI_Comm_size(MPI_COMM_WORLD, &no_process);

   //Rank of process
   int rank_process;
   MPI_Comm_rank(MPI_COMM_WORLD, &rank_process);

   // Get the name of the processor
   char processor_name[MPI_MAX_PROCESSOR_NAME];
   int name;
   MPI_Get_processor_name(processor_name, &name);

   //print message
   printf("This message is from processor: %s, rank: %d out of -  %d
processors\n",processor_name, rank_process, no_process);
//End of MPI env
MPI_Finalize();
}
```

EXECUTION:

1. mpi.h is the header file used to invoke and use all commands related MPI program.

2. MPI_Init is used to initialize MPI fields inside program.

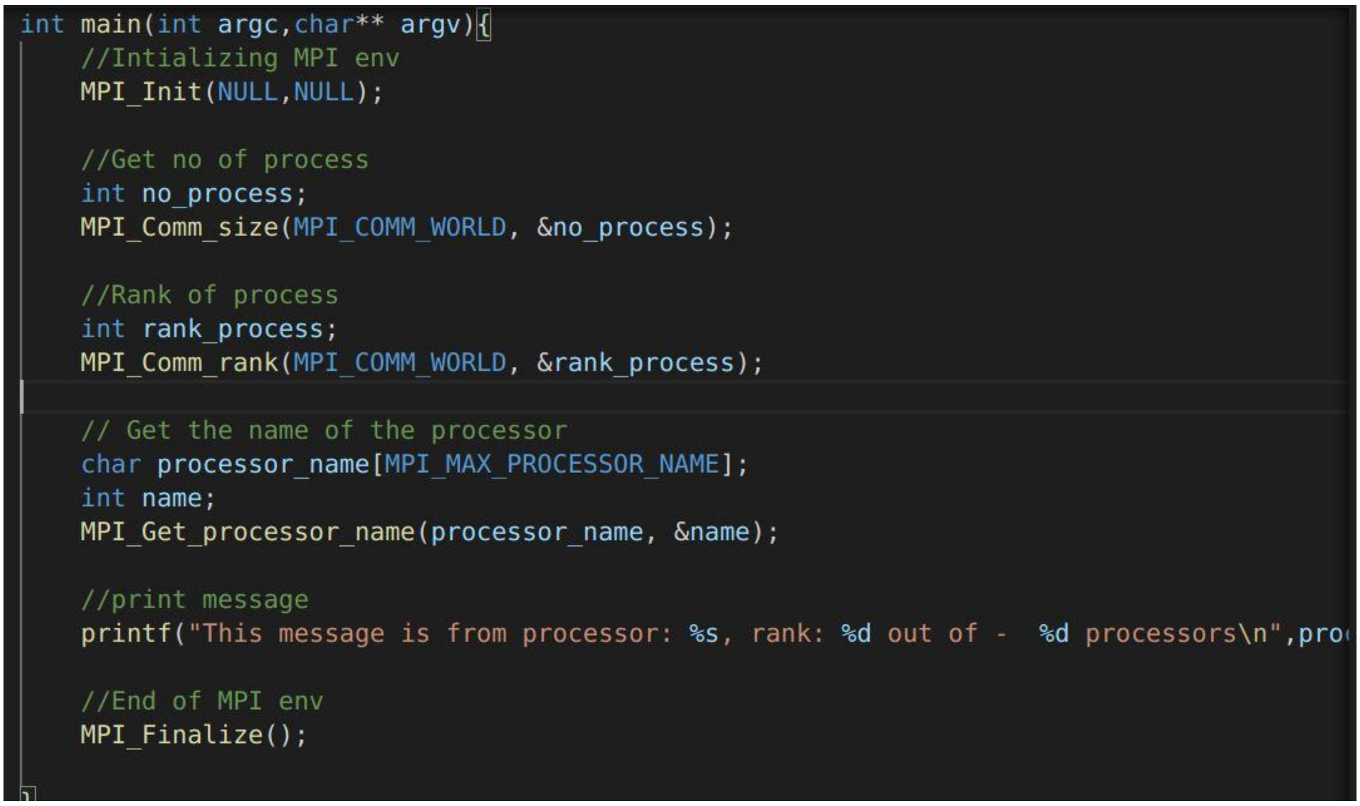
3. MPI_Comm_size gives total number of process in processors while executing program.

4. MPI_Comm_rank returns rank of current working process.

5. MPI_Get_processor_name gives current working processor name.

6. Final in printf statement, its print out message or text with all details about processors.

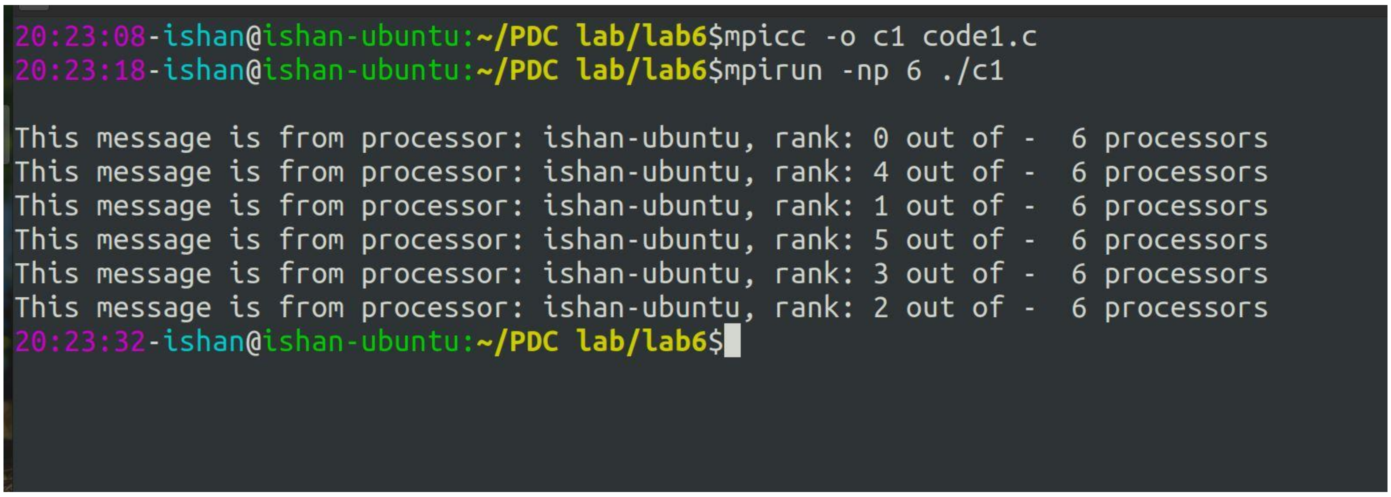
7. MPI_Finalize() is used to end MPI work environment.



```
int main(int argc,char** argv){
    //Intializing MPI env
    MPI_Init(NULL,NULL);

    //Get no of process
    int no_process;
    MPI_Comm_size(MPI_COMM_WORLD, &no_process);

    //Rank of process
    int rank_process;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank_process);

    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name;
    MPI_Get_processor_name(processor_name, &name);

    //print message
    printf("This message is from processor: %s, rank: %d out of -  %d processors\n",pro

    //End of MPI env
    MPI_Finalize();

}
```

RESULTS:

Compilation of program is done by command – mpicc and for running mpirun is used followed by -np providing total number of processes and the runnable compiled file.



```
20:23:08-ishan@ishan-ubuntu:~/PDC lab/lab6$mpicc -o c1 code1.c
20:23:18-ishan@ishan-ubuntu:~/PDC lab/lab6$mpirun -np 6 ./c1

This message is from processor: ishan-ubuntu, rank: 0 out of -  6 processors
This message is from processor: ishan-ubuntu, rank: 4 out of -  6 processors
This message is from processor: ishan-ubuntu, rank: 1 out of -  6 processors
This message is from processor: ishan-ubuntu, rank: 5 out of -  6 processors
This message is from processor: ishan-ubuntu, rank: 3 out of -  6 processors
This message is from processor: ishan-ubuntu, rank: 2 out of -  6 processors
20:23:32-ishan@ishan-ubuntu:~/PDC lab/lab6$
```

Here total no. of processes gives as 6 and compiled file is c1. It is printing message or text 6 times from each process. Rank of processes selected randomly throughout all process.