**NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**IT 301 Parallel Computing**

**LAB 7**

**28th September 2021**

**Faculty: Dr. Geetha V Dept of IT**

**Note:**

**1) For each program, you must add a screenshot of the output. Write analysis for each**

**observation.**

**2) install mpicc in ubuntu**

$ sudo apt-get install libcr-dev mpich2 mpich2-doc

**3) Steps to execute :**

mpicc helloworld.c -o hello

mpiexec -n 2 ./hello

n is the number of processes to be launched.

**MPI program 1: Simple Hello World program to find rank and size of communication world. ( 1 Mark)**

#include<mpi.h>

#include<stdio.h>

int main(int argc,char \*argv[ ])

{

int size,myrank;

**MPI\_Init(&argc,&argv);**

**MPI\_Comm\_size(MPI\_COMM\_WORLD,&size);**

**MPI\_Comm\_rank(MPI\_COMM\_WORLD,&myrank);**

printf("Process %d of %d, Hello World\n",myrank,size);

**MPI\_Finalize();**

return 0;

}

**MPI Program 2: MPI\_Send() and MPI\_Recv() for sending an integer. [Total 3 Marks]**

**(a) Note down source , destination and tag. (1 Marks)**

**(b) Modify the program to send the string “PCLAB” and add screenshot of the result. (1 marks)**

**c) Modify the program to send array of elements and add screenshot of the result. (1 marks)**

#include<mpi.h>

#include<stdio.h>

int main(int argc,char \*argv[ ])

{

int size,myrank,x,i;

MPI\_Status status;

MPI\_Init(&argc,&argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&size);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&myrank);

if(myrank==0)

{

x=10;

printf("Process %d of %d, Value of x is %d sending the value x\n",myrank,size,x);

**MPI\_Send(&x,1,MPI\_INT,1,55,MPI\_COMM\_WORLD);**

}

else if(myrank==1)

{

printf("Value of x is : %d before receive\n",x);

**MPI\_Recv(&x,1,MPI\_INT,0,55,MPI\_COMM\_WORLD,&status);**

printf("Process %d of %d, Value of x is %d\n",myrank,size,x);

printf("Source %d Tag %d \n",status.MPI\_SOURCE,status.MPI\_TAG);

}

MPI\_Finalize();

return 0;

}

**MPI Program 3: MPI\_Send() and MPI\_Recv() with MPI\_ANY\_SOURCE, MPI\_ANY\_TAG. Note down the results and write your observation. (2 Marks)**

#include<mpi.h>

#include<stdio.h>

int main(int argc,char \*argv[ ])

{

int size,myrank,x,i,y;

MPI\_Status status;

MPI\_Init(&argc,&argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&size);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&myrank);

if(myrank==0)

{

x=0;

do{

**MPI\_Recv(&x,1,MPI\_INT,MPI\_ANY\_SOURCE,MPI\_ANY\_TAG,MPI\_COMM\_WORLD,&status);**

printf("Process %d of %d, Value of x is %d : source %d tag %d error %d: \n \n",myrank,size,x,status.MPI\_SOURCE,status.MPI\_TAG,status.MPI\_ERROR);

}while(x>0);

}

else if(myrank>0)

{

y=myrank%5;

printf("Process %d of %d, Value of y is %d : sending the value y\n",myrank,size,y);

**MPI\_Send(&y,1,MPI\_INT,0,(10+myrank),MPI\_COMM\_WORLD);**

}

MPI\_Finalize();

return 0;

}

**MPI Program 4: MPI\_Send() and MPI\_Recv() with mismatched tag. Record the result for mismatched tag and also after correcting tag value of send receive as same number (2 Marks)**

#include<mpi.h>

#include<stdio.h>

int main(int argc,char \*argv[ ])

{

int size,myrank,x[50],y[50],i;

MPI\_Status status;

MPI\_Init(&argc,&argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&size);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&myrank);

printf("Verifying mistag send and receive\n");

if(myrank==0)

{

for(i=0;i<50;i++)

x[i]=i+1;

**MPI\_Send(x,10,MPI\_INT,1,10,MPI\_COMM\_WORLD);**

}

else if(myrank==1)

{

**MPI\_Recv(y,10,MPI\_INT,0,1,MPI\_COMM\_WORLD,&status);**

printf(" Process %d Recieved data from Process %d\n",myrank, status.MPI\_SOURCE);

for(i=0;i<10;i++)

printf("%d\t",y[i]);}

MPI\_Finalize();

return 0;

}

**MPI Program 5: MPI\_Send() and MPI\_Recv() standard mode:**

**Note down your observation on the content of x and y at Process 1 and Explain the importance of tag. (2 marks)**

/\* Demonstration of Blocking send and receive.\*/

#include<mpi.h>

#include<stdio.h>

int main(int argc,char \*argv[ ])

{

int size,myrank,x[10],i,y[10];

MPI\_Status status;

MPI\_Request request;

MPI\_Init(&argc,&argv);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&size);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&myrank);

if(myrank==0)

{

for(i=0;i<10;i++)

{

x[i]=1;

y[i]=2;

}

**MPI\_Send(x,10,MPI\_INT,1,1,MPI\_COMM\_WORLD);** //Blocking send will expect matching receive at the destination

//In Standard mode, Send will return after copying the data to the system buffer. The

//call will block if the buffer is not available or buffer space is not sufficient.

**MPI\_Send(y,10,MPI\_INT,1,2,MPI\_COMM\_WORLD);**

// This send will be initiated and matching receive is already there so the program will not lead to deadlock

}

else if(myrank==1)

{

**MPI\_Recv(x,10,MPI\_INT,0,2,MPI\_COMM\_WORLD,&status);**

//P1 will block as it has not received a matching send with tag 2

for(i=0;i<10;i++)

printf("Received Array x : %d\n",x[i]);

**MPI\_Recv(y,10,MPI\_INT,0,1,MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);**

for(i=0;i<10;i++)

printf("Received Array y : %d\n",y[i]);

}

MPI\_Finalize();

return 0;

}