# Department of Computing

**CS332: Distributed Computing**

**Class: BSCS – 6 & BESE – 7**

# Lab 12: Message Passing Interface (MPI)

**Date: 23rd December, 2019**

**Time: 09:00 AM – 12:00 PM**

# Instructor: Mr. Suleman Awan

**Lab Engineer: Mr. Ahsan Gul**

**Name: Ahmad Amjad Mughal**

**Reg No: 121672**

**Class: BSCS-6C**

# Lab 12: Message Passing Interface (MPI)

**Introduction**

This lab introduces hands on experience to the Implement the leader selection algorithm using Message Passing Interface (MPI)

**Objectives**

Understand and implement using Message Passing Interface (MPI).

**Tools/Software Requirement**

MS Visual Studio 2013

**Description**

Implement the following leader selection algorithm using Message Passing Interface.

Algorithm

Let Pi be the processor starting the election   
(it detected e.g. that the current leader has crashed).   
Pi sends a message (i,*elect*) to its right neighbor (direction of the ring !) in the ring.   
Pj receives a (i,*elect*) :   
if i = j then   
    Pi is a new coordinator, send (i, *Iamtheboss*) to its neighbor.   
else   
    Pj sends (max(i,j),*elect*) to its right neighbor

1. To setup MPI environment in MS Visual Studio 2012, install HPC Pack 2012 MS-MPI redistribution from the following link <https://www.microsoft.com/en-us/download/details.aspx?id=57467>
2. Create an empty Visual C++ Win 32 console Application
3. Uncheck the Pre-compiled headers and Security Development Lifecycle checkboxes
4. Click on empty project check box
5. Right click on your project to add C++ source file
6. Go to project properties to add the additional include and lib files folder
   1. Under C++ tag, select the General tag – Additional Include Directories and add the path to **C:\Program Files 28x8629\Microsoft SDKs\MPI\Include\x86**

**[check the path in your system]**

* 1. Click on Linker tab under properties – General – Additional Libraries and browse to lib folder i.e. **C:\Program Files 28x8629\Microsoft SDKs\MPI\Lib\x86**

**[Check the path on your system]**

* 1. Remember to include the x86 architecture
  2. Again click on Linker – Input – Additional Dependencies and add **“msmpi.lib”**
  3. **Press apply and write the below mentioned code. It should run with default number of processes.**

#include "stdafx.h"

#include <iostream>

#include "mpi.h"

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

{

//cout << "Hello World\n";

int nTasks, rank;

MPI\_Init(&argc,&argv);

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

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

printf ("Number of threads = %d, My rank = %d\n", nTasks, rank);

int a;

std::cin>>a;

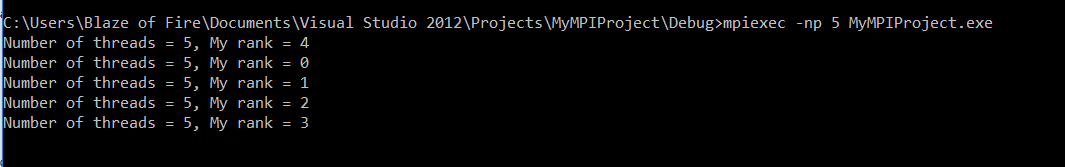
MPI\_Finalize();

return 0;

}

After compiling the code, you can also run from the command line using following command.

1. Open command prompt, CD to the directory where .exe file is place
2. Type mpiexec –np 5 text.exe
3. It will display following output



**Lab Tasks**

Your task is to implement the above Leader Selection Algorithm using Message Passing Interface (MPI).

**Task Solution**

//Header files and Pre Processor directories

#include "mpi.h"

#include<iostream>

#include<vector>

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

using namespace std;

//main function

int main()

{

//defined variables

int myrank,nprocs,namelen;

int initiator=0;

char processorName[10];

MPI\_Init(NULL,NULL);

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

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

MPI\_Get\_processor\_name(processorName, &namelen);

//to find the maximum rank

int max=0;

int \*array = new int[nprocs];

// int array[nprocs];

//initialize all the processes with -1

for(int i=0;i<nprocs;++i)

{

array[i]=-1;

}

int root=0;

if(myrank==0)

{

while(initiator==0)

{

srand(time(NULL));

initiator=rand()%nprocs;

}

//initializer starts messaging and sends a message to process of higher rank

cout<<"The inititaor for all the processses will be "<<initiator<<endl;

}

MPI\_Bcast(&initiator,1,MPI\_INT,0,MPI\_COMM\_WORLD);

if(myrank == initiator)

{

cout<<"\nThe initiator is(IN THE INITIATOR) "<<initiator<<endl;

array[myrank] = myrank;

//send to the process which is 1 bigger means make a ring

MPI\_Send(array,nprocs,MPI\_INT,(myrank+1)%nprocs,999,MPI\_COMM\_WORLD);

cout<<"\n\t\t"<<myrank <<" is waiting for the last process "<<(myrank-1)%nprocs<<" running on "<<processorName<<endl<<endl;

MPI\_Recv(array, nprocs, MPI\_INT, (myrank-1)%nprocs, 999, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

cout<<"\n\t\tThe initiator received the message \n"<<endl;

cout <<"My rank is "<< myrank <<" running on "<<processorName<< endl<<endl;

cout<<"The array in the rank "<<myrank<<" is"<<endl;

for(int j=0;j<nprocs;++j)

{

cout<<array[j]<<" ";

}

cout<<endl;

max = 0;

for(int i=0;i<nprocs;++i)

{

if(max<array[i])

max = array[i];

}

cout<<"Winner is "<<max<<endl;

//Broadcasting winner of the election

}

else if(myrank!=initiator){

if(myrank == 0)

{

MPI\_Recv(array, nprocs, MPI\_INT, (nprocs-1)%nprocs, 999, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

cout<<" My rank is "<<myrank<<" and I have received from "<<(nprocs-1)%nprocs<<" running on "<<processorName<<endl;

}

else

{

MPI\_Recv(array, nprocs, MPI\_INT, (myrank-1)%nprocs, 999, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

cout<<" My rank is "<<myrank<<" and I have received from "<<(myrank-1)%nprocs<<" running on "<<processorName<<endl;

}

cout<<"The array in the rank "<<myrank<<" is"<<endl;

for(int i=0;i<nprocs;++i)

{

cout << array[i] << " ";

}

srand(time(NULL));

int random= rand()%2+1;

//Means he wants to participitate 2 or yes and 1 for no

if(random==2)

array[myrank] = myrank;

cout<<"\nSending array to the rank "<<(myrank+1)%nprocs<<endl<<endl;

MPI\_Send(array,nprocs,MPI\_INT,(myrank+1)%nprocs,999,MPI\_COMM\_WORLD);

}

MPI\_Bcast(&max,1,MPI\_INT,max,MPI\_COMM\_WORLD);

cout<<myrank<<" received thank you from winner \n";

MPI\_Finalize();

}

**Screenshot**

