

Fall 2025

Riphah International University, Islamabad

Faculty of Computing

Faculty of Computing



Lab No 13

MS MPI

Fall 2025
Islamabad

Riphah International University,

Faculty of Computing

Lab Manual Development Team

Supervision and Coordination

Dr. Hajra Murtaza

Assistant Professor

Faculty of Computing

Lab Designers

Ayesha Kousar

Lab Instructor

Faculty of Computing

Lab 13: MS MPI

1. Introduction

In this lab, we explore **MS MPI**. Students will implement practical tasks to understand:

- What a MS MPI is
- What MS MPI are used for
- How MS MPI work
- Real life applications of MS MPI

3. Activity Time Boxing

Task No.	Activity Name	Activity Time	Total Time
1	Conceptual Overview	10 min	
2	MS MPI Implementation	30 mins	
3	Walk through Task	30 mins	
5	Lab Exercise & Demo	80 mins	150 mins

4. Key Concepts & Syntax

Microsoft MPI (MS MPI) is a Microsoft implementation of the Message Passing Interface standard for developing and running parallel applications on the Windows platform.

MS MPI offers several benefits:

- Security based on Active Directory Domain Services.
- High performance on the Windows operating system.
- Binary compatibility across different types of interconnectivity options.

MS MPI Downloads

The following are current downloads for MS MPI:

[Download Microsoft MPI v10.1.3 from Official Microsoft Download Center](#)

Download MS MPI execution file and Sdk

Requirements:

Visual Studio (with C++ Distribution)

MS MPI , SDK

Steps for running a MS MPI Program on two nodes

- Create a New C++ project

- New Project -> Windows Desktop Application
The name of the project and files should be the same for two machines.

- Write a Basic program of MS MPI which displays total processes, the current running process and processor name
Here is the sample code:
Insert code in c++ file:

```
#include <iostream>
#include <mpi.h>
#include <vector>

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

    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    int my_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    std::cout << "Process " << my_rank << " of " << world_size
        << " is running on machine: " << processor_name << std::endl;

    MPI_Finalize();

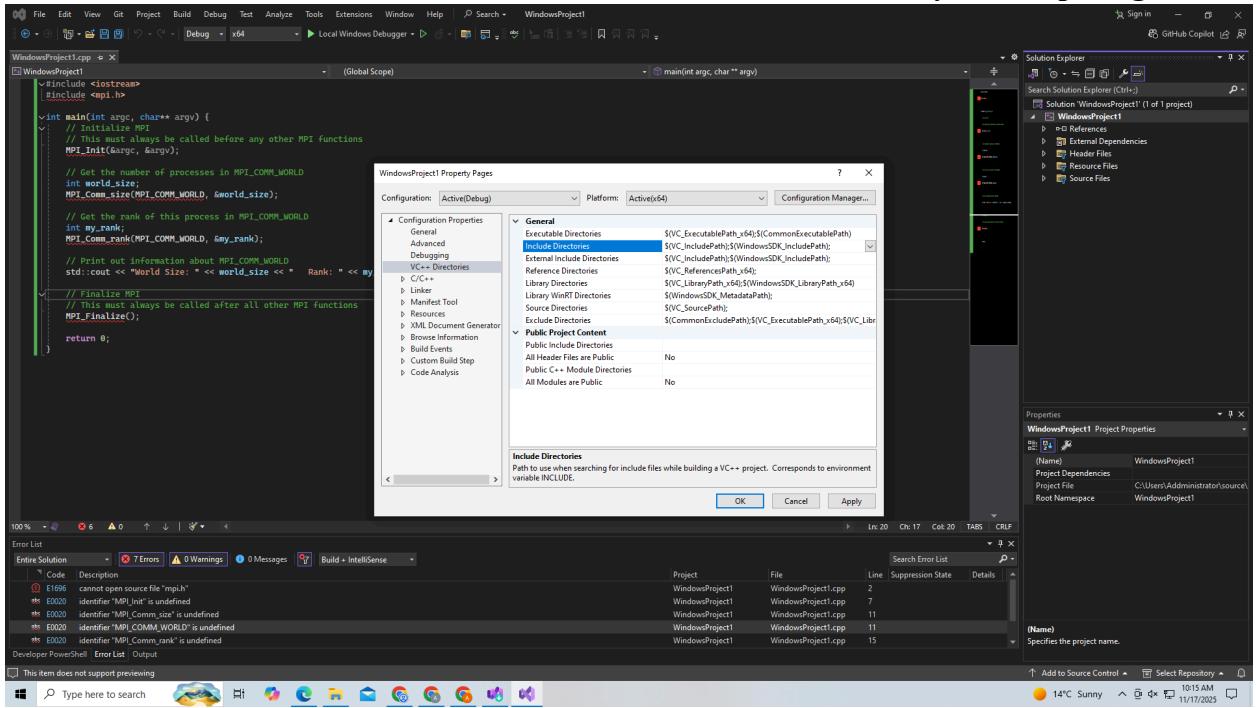
    return 0;
}
```

- Add paths of MS MPI in you project properties as given below:

Fall 2025

Riphah International University, Islamabad

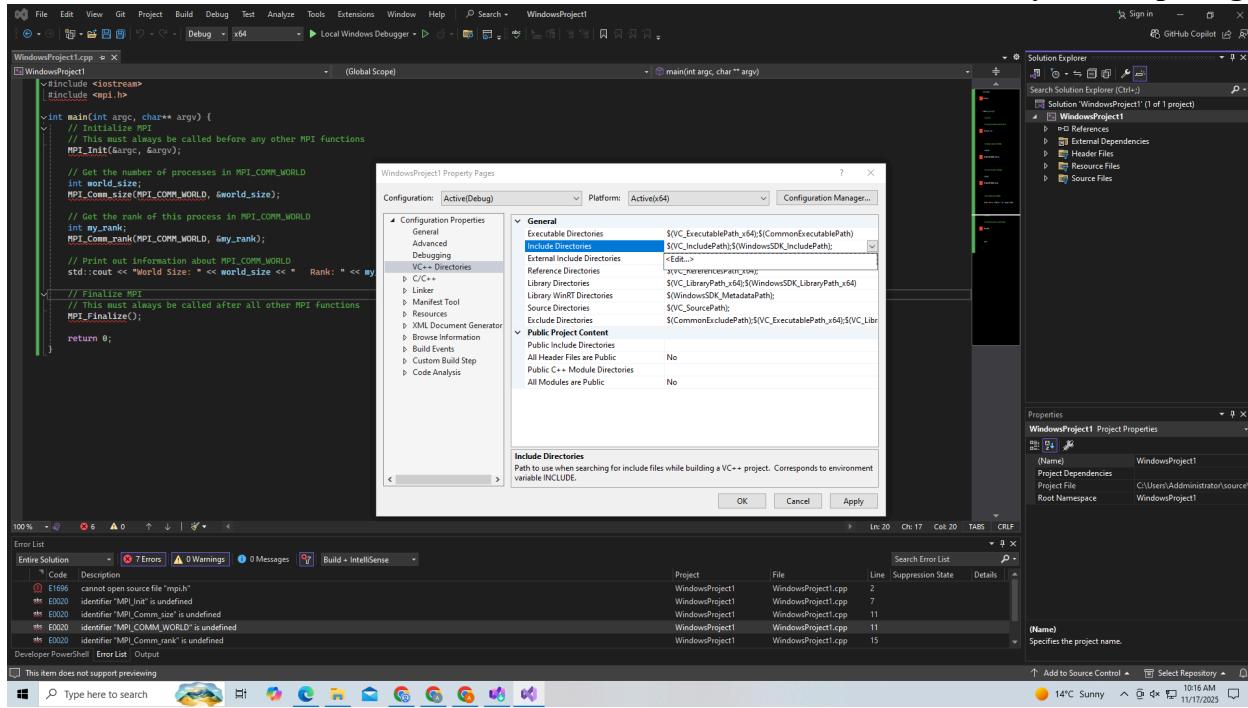
Faculty of Computing



Fall 2025

Riphah International University, Islamabad

Faculty of Computing



- These are the paths (verify your installation by checking the folder directory)

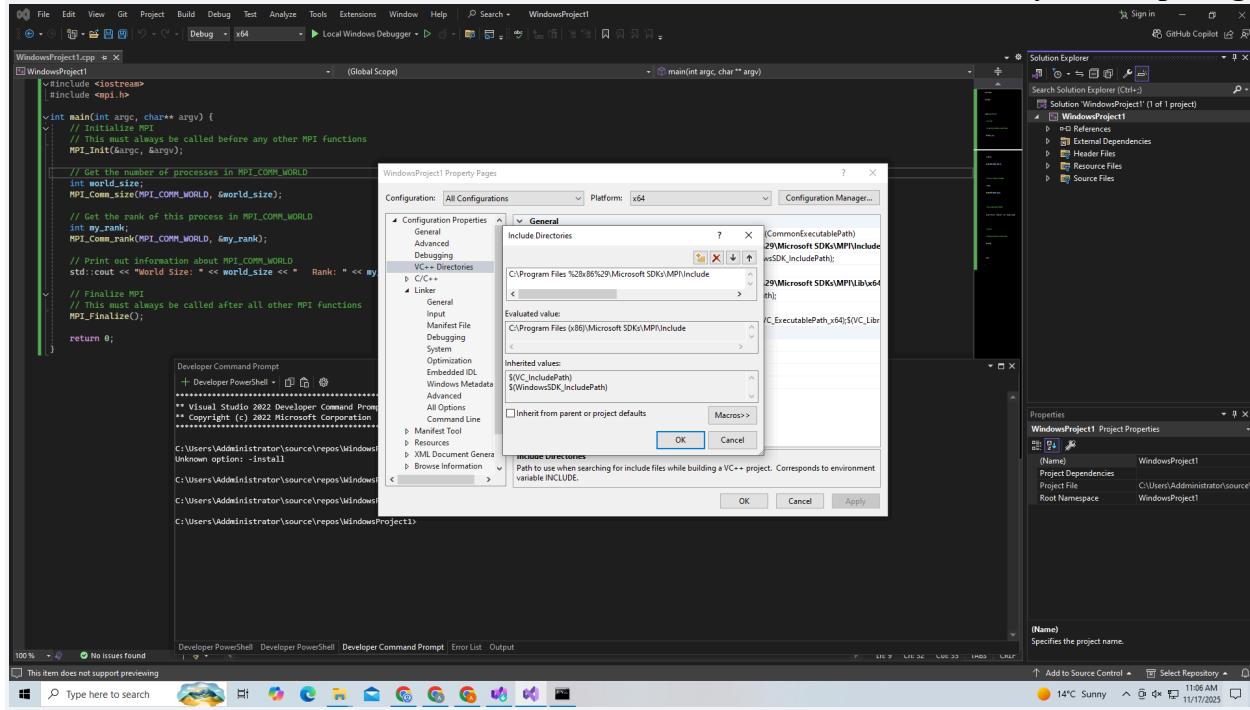
C:\Program Files (x86)\Microsoft SDKs\MPI\Include

C:\Program Files (x86)\Microsoft SDKs\MPI\Lib\x64

Fall 2025

Riphah International University, Islamabad

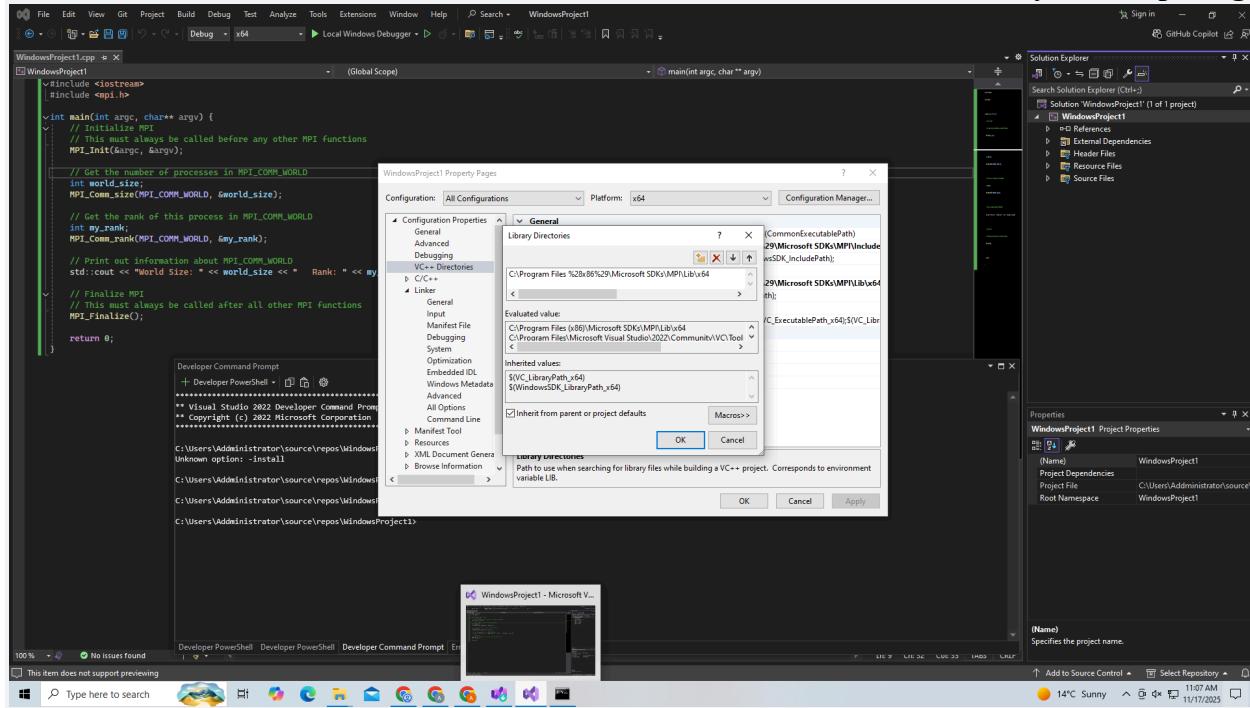
Faculty of Computing



Fall 2025

Riphah International University, Islamabad

Faculty of Computing



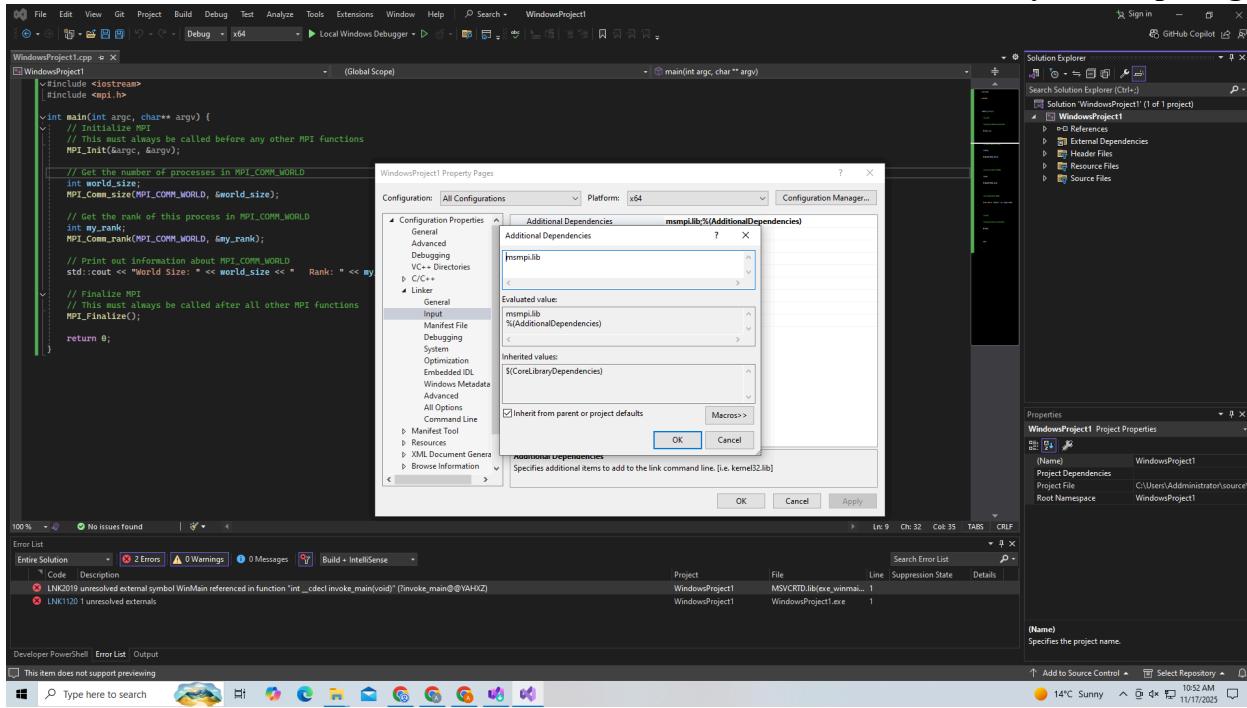
- In Additional directories insert

msmpi.lib

Fall 2025

Riphah International University, Islamabad

Faculty of Computing

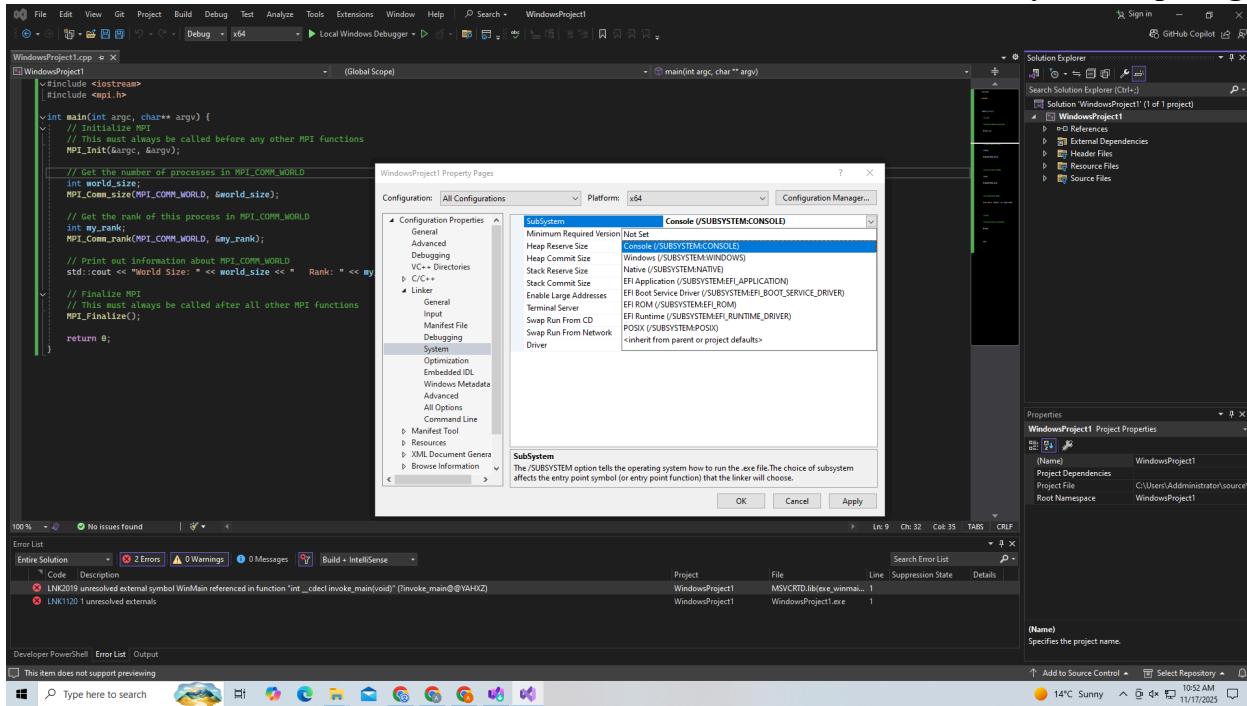


- In linker systems , choose subsystem and from dropdown select subsystem console:

Fall 2025

Riphah International University, Islamabad

Faculty of Computing



- Execution Command:

Mpiexec -np 4 **project_name.exe**

Fall 2025

Riphah International University, Islamabad

Faculty of Computing

```
WindowsProject1.cpp  x
WindowsProject1
main.cpp
WindowsProject1.h
WindowsProject1.vcxproj
WindowsProject1.vcxproj.filters
WindowsProject1.sln

WindowsProject1.cpp
#include <iostream>
#include <mpi.h>

int main(int argc, char** argv) {
    // Initialize MPI
    // This must always be called before any other MPI functions
    MPI_Init(&argc, &argv);

    // Get the number of processes in MPI_COMM_WORLD
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of this process in MPI_COMM_WORLD
    int my_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    // Print out information about MPI_COMM_WORLD
    std::cout << "World Size: " << world_size << " Rank: " << my_rank << std::endl;

    // Finalize MPI
    // This must always be called after all other MPI functions
    MPI_Finalize();

    return 0;
}

Developer PowerShell
+ Developer PowerShell + ① ② ③
PS C:\Users\Administrator\source\repos\WindowsProject1\x64\Debug> mpexec -np 4 WindowsProject1.exe
>>>
World Size: 4  Rank: 3
World Size: 4  Rank: 2
World Size: 4  Rank: 1
World Size: 4  Rank: 0
PS C:\Users\Administrator\source\repos\WindowsProject1\x64\Debug>
```

- By using Command prompt (Administrative), turn off firewall.

Command:

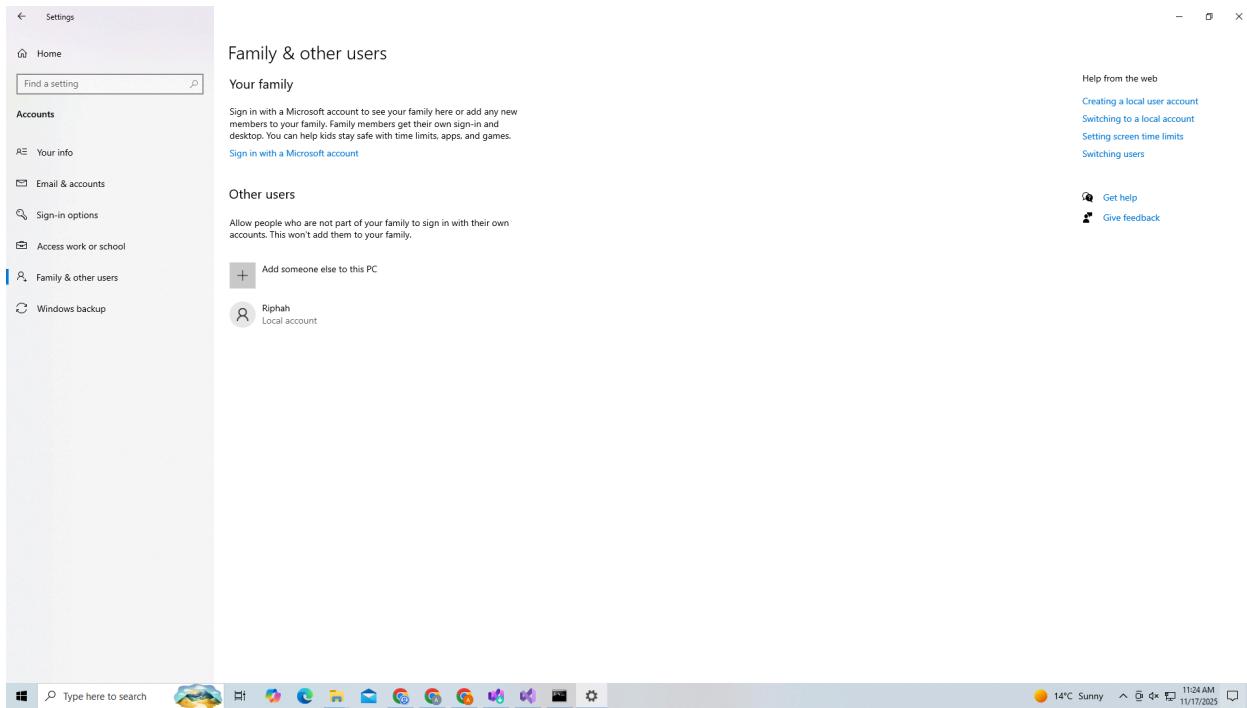
```
netsh advfirewall set allprofiles state off
```

Note: Smpd is a cluster process manager

Fall 2025

Riphah International University, Islamabad
Faculty of Computing

- For running on multiple machines, two machines should be logged in the same accounts or should do smpd settings.



- Use ping command to check connection:

```
C:\> Administrator: Command Prompt
C:\Program Files\Microsoft MPI\Bin>smpd -phrase
Unknown option: -phrase

C:\Program Files\Microsoft MPI\Bin>hostname
SystemLab1-47

C:\Program Files\Microsoft MPI\Bin>ping MachineSystemLab1-44
Ping request could not find host MachineSystemLab1-44. Please check the name and try again.

C:\Program Files\Microsoft MPI\Bin>ping SystemLab1-44

Pinging systemLab1-44.local [fe80::3653:12b8:f765:d97e%4] with 32 bytes of data:
Reply from fe80::3653:12b8:f765:d97e%4: time=2ms
Reply from fe80::3653:12b8:f765:d97e%4: time=2ms
Reply from fe80::3653:12b8:f765:d97e%4: time=2ms
Reply from fe80::3653:12b8:f765:d97e%4: time=2ms

Ping statistics for fe80::3653:12b8:f765:d97e%4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 2ms, Average = 2ms

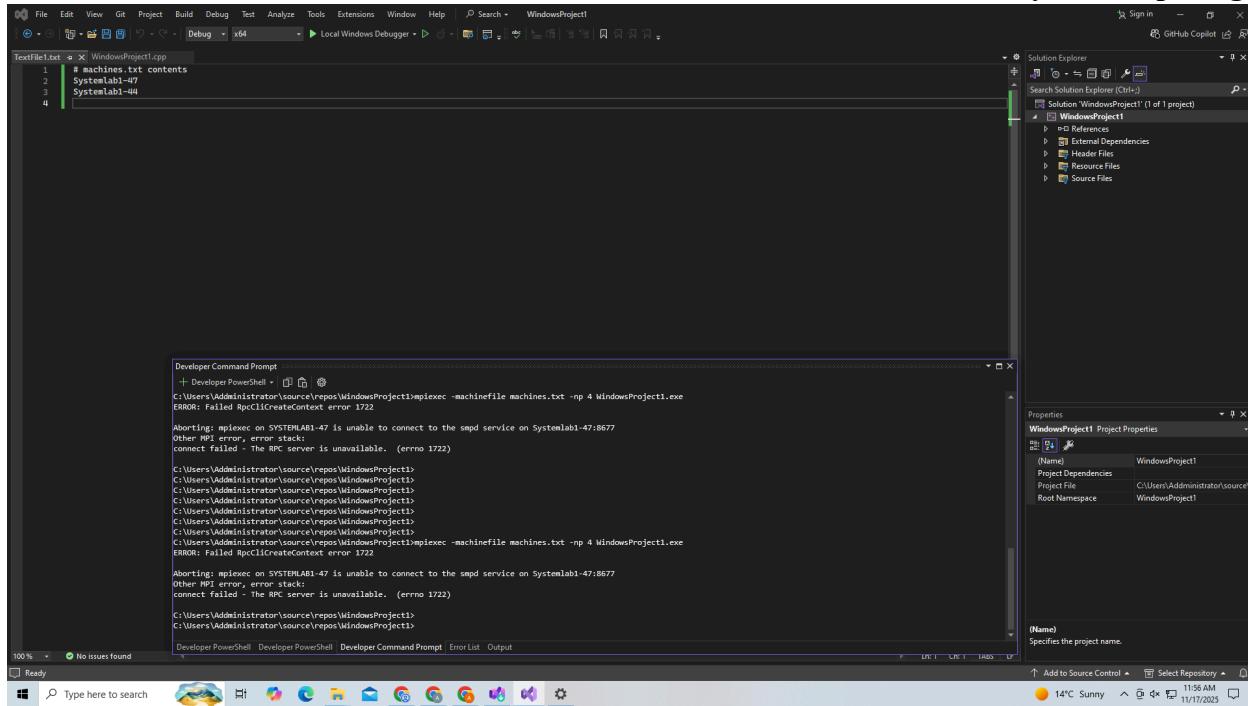
C:\Program Files\Microsoft MPI\Bin>
C:\Program Files\Microsoft MPI\Bin>
```

- Create a machine file (txt) to run it on multiple machines, Add machines host names
e.g SystemLab-14

Fall 2025

Riphah International University, Islamabad

Faculty of Computing

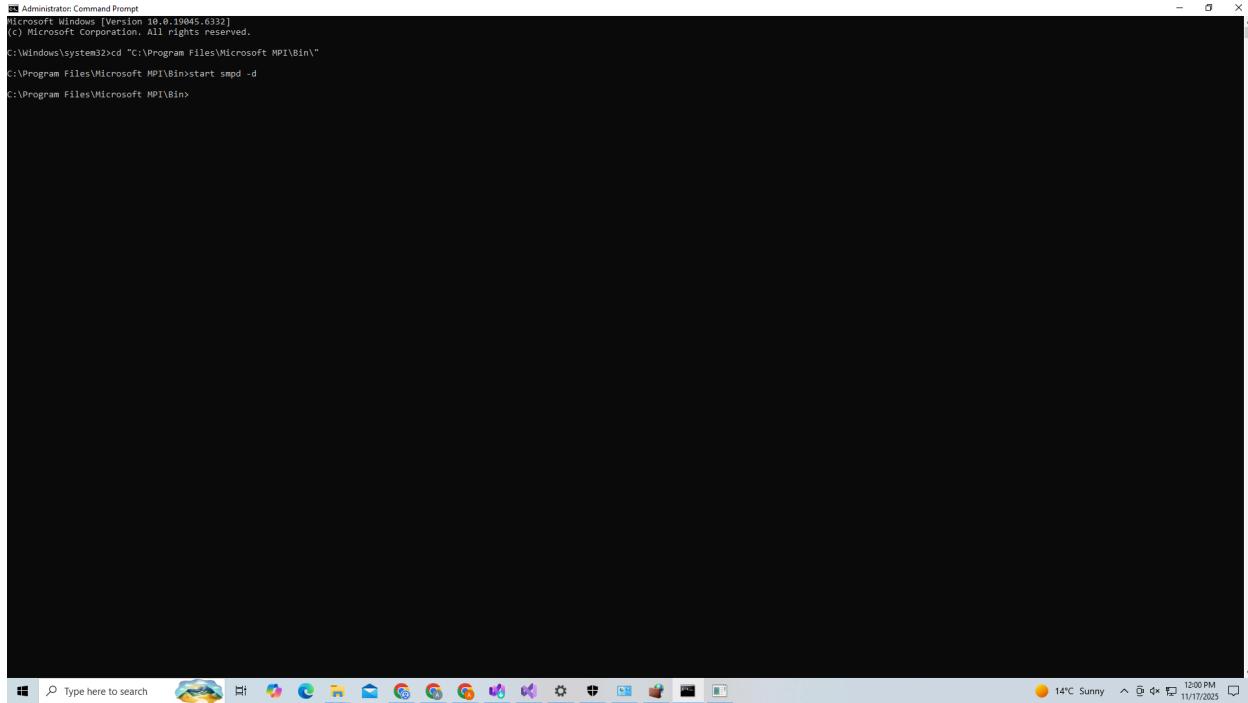


- Command for starting smpd settings on all machines.
cd "C:\Program Files\Microsoft MPI\Bin"
start smpd -d

Fall 2025

Riphah International University, Islamabad

Faculty of Computing



A screenshot of a Windows operating system desktop. In the center is a command prompt window titled "Administrator: Command Prompt". The window contains the following text:

```
Microsoft Windows [Version 10.0.10645.6332]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd "C:\Program Files\Microsoft MPI\Bin"
C:\Program Files\Microsoft MPI\Bin>start smpd -d
C:\Program Files\Microsoft MPI\Bin>
```

The desktop background is black. At the bottom, the Windows taskbar is visible with various pinned icons. On the right side of the taskbar, there is weather information showing "14°C Sunny" and a date/time stamp "12:00 PM 11/17/2023".

- Command for execution using different machines
Mpexec -machinefile **machines.txt** -np 4 **project_name.exe**

The screenshot shows the Microsoft Visual Studio IDE interface. On the left is the code editor with C++ code for MPI communication. In the center, a terminal window displays the output of the MPI program, showing 16 processes running on a machine named 'systemlab1-47'. The output includes process IDs, ranks, and world sizes. On the right are the Solution Explorer, Properties, and Task List panes.

```

machines.txt WindowsProject1.cpp > X
WindowsProject1 (Global Scope) main(int argc, char ** argv)
#include <iostream>
#include <mpi.h>
#include <vector> // Use vector for character buffer as good practice

int main(int argc, char** argv) {
    // Initialize MPI
    MPI_Init(&argc, &argv);

    // Get the number of processes in MPI_COMM_WORLD
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of this process in MPI_COMM_WORLD
    int my_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    // Get the processor name
    // MPI_MAX_PROCESSOR_NAME is a predefined constant for the max name length
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Print out information about MPI_COMM_WORLD including the processor name
    std::cout << "Process " << my_rank << " of " << world_size
    << " is running on machine: " << processor_name << std::endl;

    // Finalize MPI
    MPI_Finalize();
    return 0;
}

World Size: 16 Rank: 15
World Size: 16 Rank: 9
World Size: 16 Rank: 1
World Size: 16 Rank: 3
PS C:\Users\Administrator\source\repos\WindowsProject1\x64\Debug>
>> mpiexec -n 16 machines
Process 14 of 16 is running on machine: systemlab1-47
Process 0 of 16 is running on machine: systemlab1-47
Process 5 of 16 is running on machine: systemlab1-47
Process 12 of 16 is running on machine: systemlab1-47
Process 10 of 16 is running on machine: systemlab1-47
Process 6 of 16 is running on machine: systemlab1-47
Process 9 of 16 is running on machine: systemlab1-47
Process 11 of 16 is running on machine: systemlab1-47
World Size: 16 Rank: 7
World Size: 16 Rank: 9
World Size: 16 Rank: 15
World Size: 16 Rank: 5
World Size: 16 Rank: 11
World Size: 16 Rank: 1
World Size: 16 Rank: 3
World Size: 16 Rank: 13
PS C:\Users\Administrator\source\repos\WindowsProject1\x64\Debug>

```

Communication ways in MS MPI:

Point to point Communication:

- **MPI_Send** → Sends a message and blocks until the send buffer is safe to modify.
- **MPI_Recv** → Receives a message and blocks until the data arrives.

Collective Communication:

Data movement

- **MPI_Bcast** – Broadcasts data from one process to all others in a communicator.
- **MPI_Scatter** – Distributes distinct chunks of an array from root to all processes.

- **MPI_Scatterv** – Scatter with variable-sized chunks for each process.
- **MPI_Gather** – Collects equal-sized data blocks from all processes to the root.
- **MPI_Gatherv** – Gather with variable-sized blocks from each process.
- **MPI_Allgather** – Every process receives the gathered data from all processes.
- **MPI_Allgatherv** – Allgather version supporting variable block sizes.
- **MPI_Alltoall** – Each process sends a distinct block to every other process.
- **MPI_Alltoallv** – Like Alltoall but with variable block sizes.

Reductions

- **MPI_Reduce** – Applies a reduction operation (sum, max, etc.) and returns the result to root.
- **MPI_Allreduce** – Reduction whose result is returned to all processes.
- **MPI_Reduce_scatter** – Reduces data and scatters the resulting segments to each process.
- **MPI_Scan** – Performs a prefix reduction across processes.
- **MPI_Exscan** – Exclusive prefix reduction (process 0 receives no value).

Synchronization

- **MPI_Barrier** – All processes wait until every member reaches the barrier.

LAB TASK:

Write an MPI program that performs parallel matrix vector multiplication using multiple processes. The root process should read or generate an $N \times N$ matrix A and a vector x .

Fall 2025

Riphah International University, Islamabad

Faculty of Computing

It must then scatter the rows of the matrix to all worker processes. Each process computes its portion of the output vector $y = Ax$. After computation, all partial results must be gathered at the root form the final result vector.