Name: Muhammad Allah Rakha

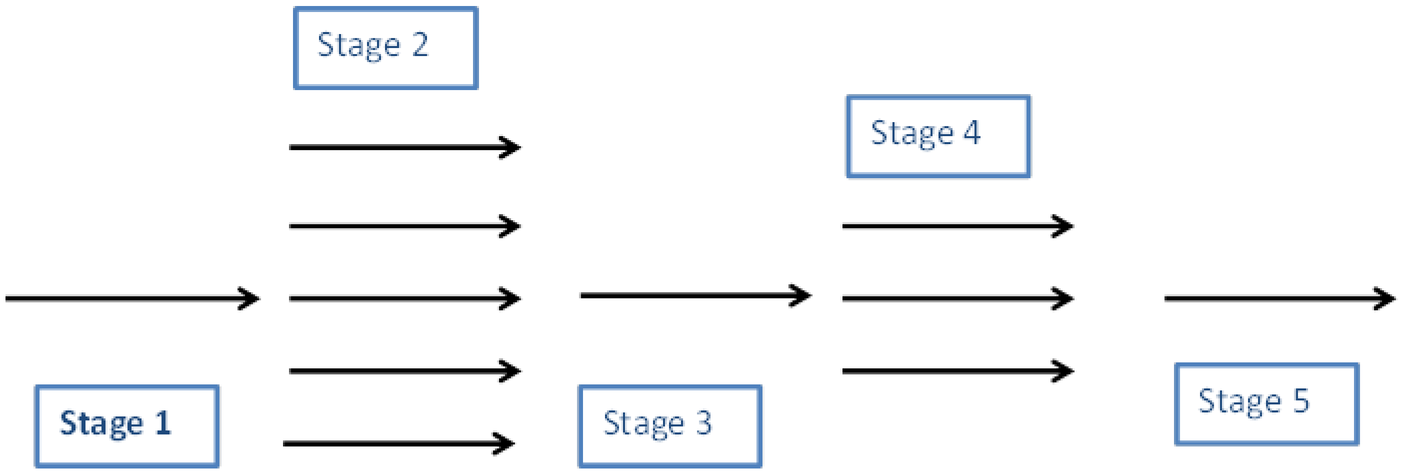
Roll No: P19-0006 BCS-6A (PDC)

Assignment: MPI and Open-MP

A Brief about MPI & Open MP

1. MPI stands for *Message Passing Interface.* These are available as API (Application programming interface) or in library form for C, C++ and FORTRAN. Different MPI's API are available in market i.e. OpenMPI, MPICH, HP-MPI.Intel MPI, etc. Whereas many are freely available like OpenMPI, MPICH etc., other like Intel MPI comes with license i.e. you need to pay for it. One can use any one of above to parallelize programs. MPI standards maintain that all of these APIs provided by different vendors or groups follow similar standards, so all functions or subroutines in all different MPI API follow similar functionality as well arguments. The difference lies in implementation that can make some MPIs API to be more efficient than other. Many commercial CFD-Packages gives user option to select between different MPI API. However HP-MPI as well Intel MPIs are considered to be more efficient in performance. When MPI was developed, it was aimed at distributed memory system but now focus is both on distributed as well shared memory system. However it does not mean that with MPI, one cannot run program on shared memory system, it just that earlier, we could not take advantage of shared memory but now we can with latest MPI 3.

2. Open MP stand for *Open Multiprocessing.* Open MP is basically add on in compiler. It is available in gcc (gnu compiler), Intel compiler and with other compilers. Open MP target shared memory systems i.e. where processor shared the main memory. Open MP is based on thread approach. It launches a single process which in turn can create *n* number of thread as desired. It is based on what is called "fork and join method" i.e. depending on particular task it can launch desired number of thread as directed by user.

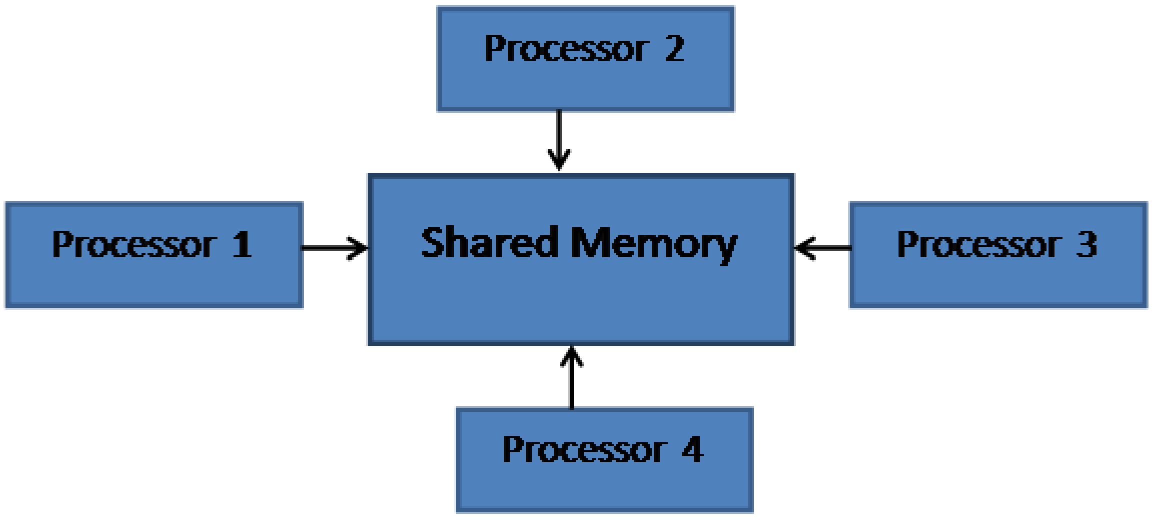


Fork and Join Model of Open MP: Different stage of program show different number of thread

Programming in Open MP is relatively easy and involve adding *pragma* directive. User need to tell number of thread it need to use. (Note that launching more thread than number of processing unit available can actually slow down the whole program)

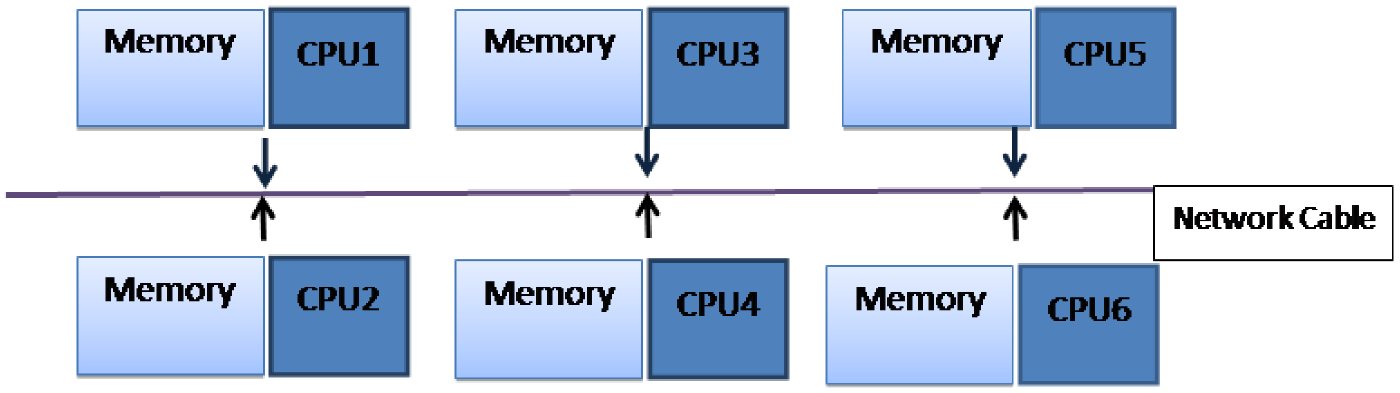
What is Shared Memory and Distributed Memory?

1) Shared memory is one where all processors can see whole of the memory that is available. Simple example is your desktop computer or laptop, where all processing units can see all the memory of system



Shared Memory: Processor 1, 2 3 4 can see whole memory

2) Distributed memory system is one where processor can see limited memory i.e. two desktop computer connected in network. They can only see memory available to them only not of other



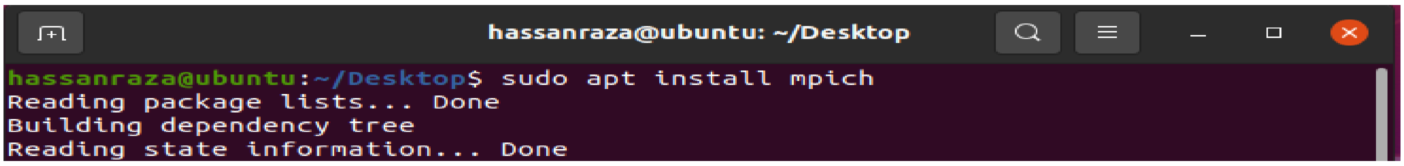
Distributed Memory System: CPU can see only limited memory of their own

What is Process and Thread?

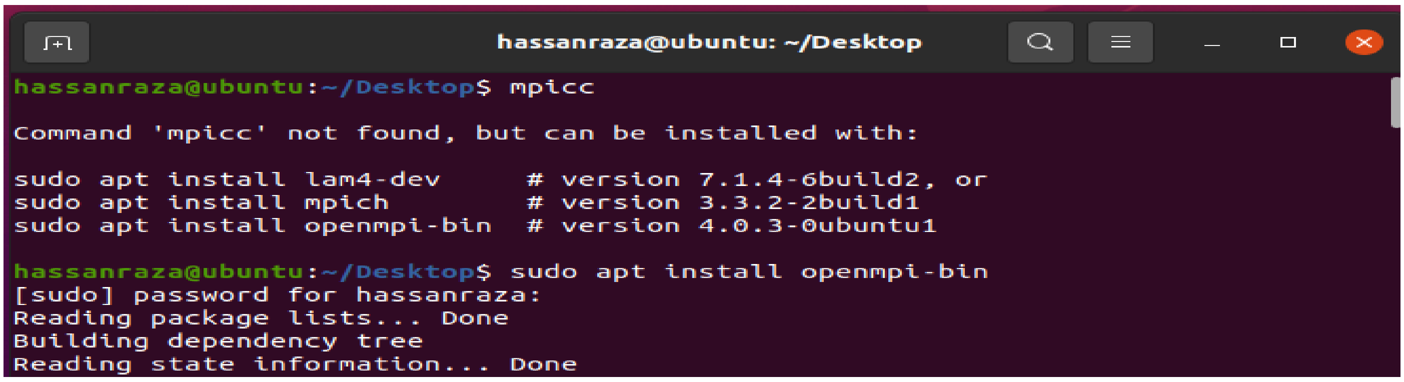
1) Process: An executing instance of program .It has distinct address space. It different from other executing instance of program in way that it has separate resources.

2) Thread is subset of process. A process can have *n* number of threads as desired. Every thread of process share it's all resources i.e. data as well address space of process that created it. Thread has to be part of some process. It cannot be independent.

MPI	Open MP
Available from different vendor and can be compiled in desired platform with desired compiler. One can use any of MPI API i.e. MPICH, OpenMPI or other	Open MP are hooked with compiler so with gnu compiler and with Intel compiler one have specific implementation. User is at liberty with changing compiler but not with Open MP implementation.
• MPI support C,C++ and FORTRAN • OpenMPI one of API for MPI is providing provisional support for Java • MPI target both distributed as well shared memory system • Process in MPI has private variable only, no shared variable	• Open MP support C,C++ and FORTRAN • Few projects try to replicate Open MP for Java. • Open MP target only shared memory system • In Open MP , threads have both private as well shared variable
Compilation of MPI program require. 1. Adding header file: #include "mpi.h" 2. compiler as:(in Linux) GCC and MPICH2 for MPI • **mpic++ mpi.cxx -o mpiExe** (User need to set environment variable PATH and LD_LIBRARY_PATH to MPI as OpenMPI installed folder or binaries) (For Linux)	Need to add omp.h and then can directly compile code with -fopenmp in Linux environment. GCC-4.2 with library libgomp for OpenMP • g++ -fopenmp openmp.cxx -o openmpExe
Running MPI program. a) User need to make sure that bin and library folder from MPI installation are included in environmental variable PATH and LD_LIBRARY_PATH. b) For running executable from command line ,user need to supply following command and specify number of processor as in example below it is four	User can launch executable *openmpExe* in normal way
mpirun -np 4 mpiExe	./openmpExe

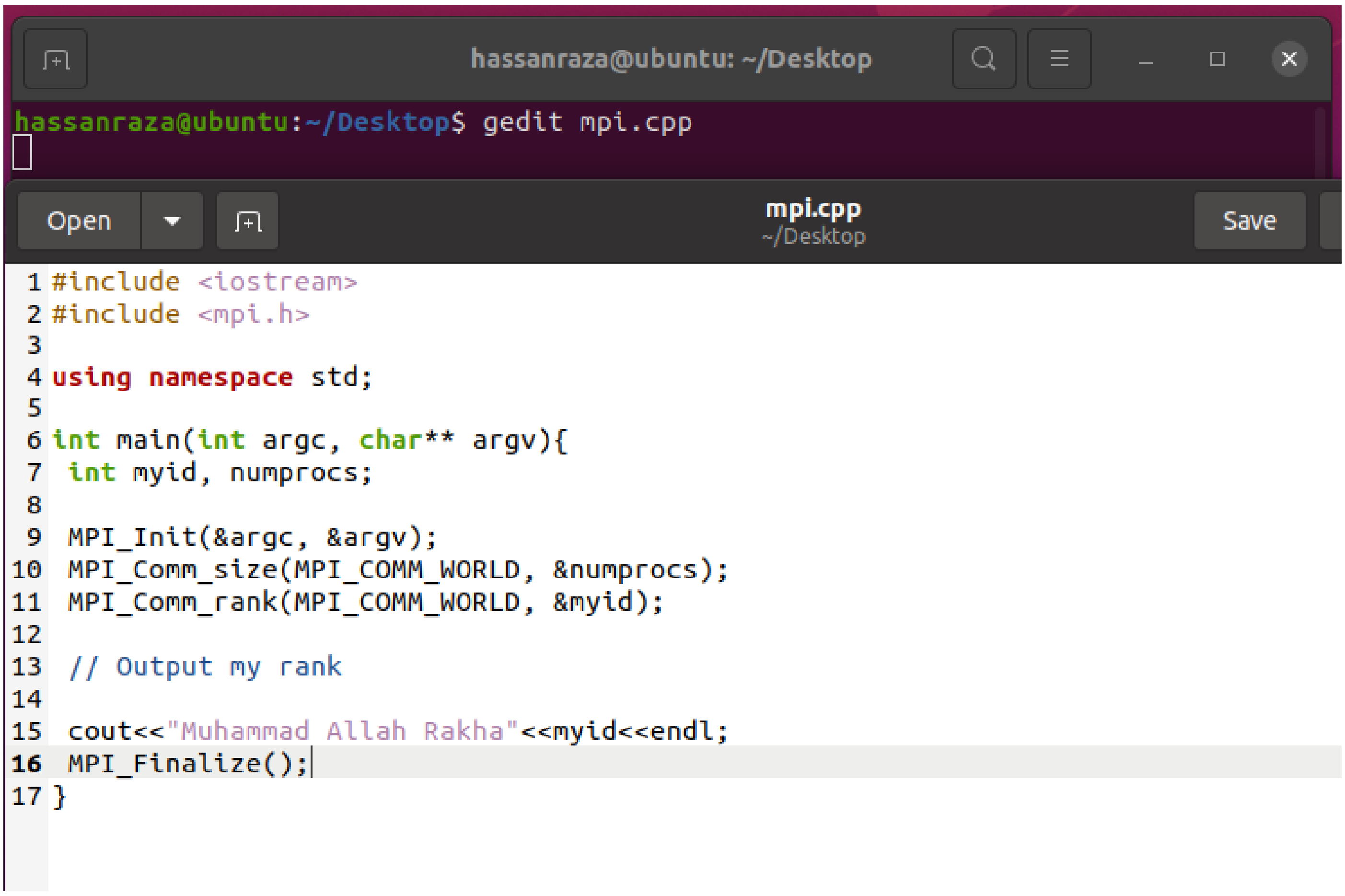


```
hassanraza@ubuntu:~/Desktop$ sudo apt install mpich
Reading package lists... Done
Building dependency tree
Reading state information... Done
```



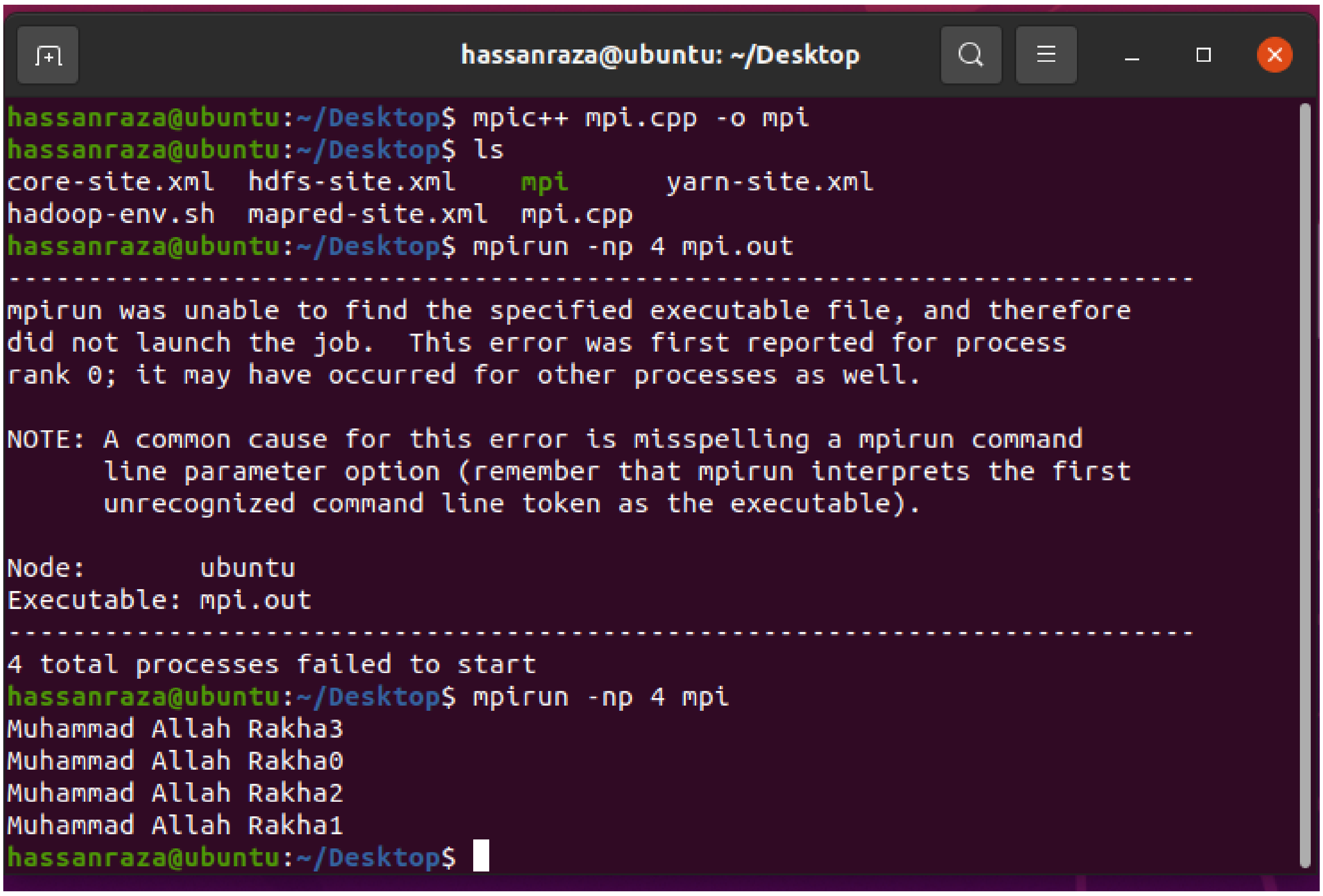
```
hassanraza@ubuntu:~/Desktop$ mpicc

Command 'mpicc' not found, but can be installed with:

sudo apt install lam4-dev      # version 7.1.4-6build2, or
sudo apt install mpich         # version 3.3.2-2build1
sudo apt install openmpi-bin   # version 4.0.3-0ubuntu1

hassanraza@ubuntu:~/Desktop$ sudo apt install openmpi-bin
[sudo] password for hassanraza:
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

MPI



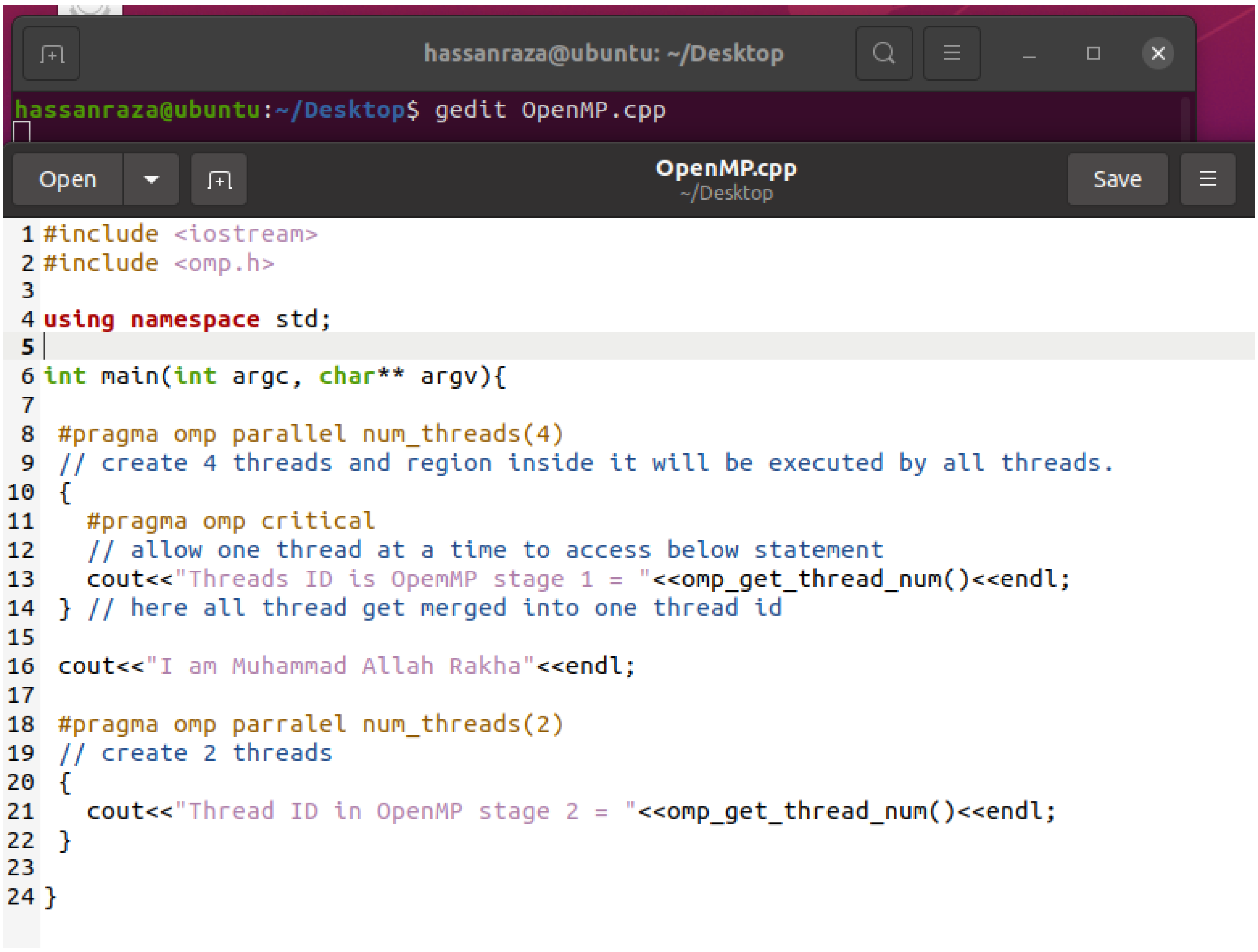
```
hassanraza@ubuntu:~/Desktop$ gedit mpi.cpp
```

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

using namespace std;

int main(int argc, char** argv){
 int myid, numprocs;

 MPI_Init(&argc, &argv);
 MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
 MPI_Comm_rank(MPI_COMM_WORLD, &myid);

 // Output my rank

 cout<<"Muhammad Allah Rakha"<<myid<<endl;
 MPI_Finalize();
}
```



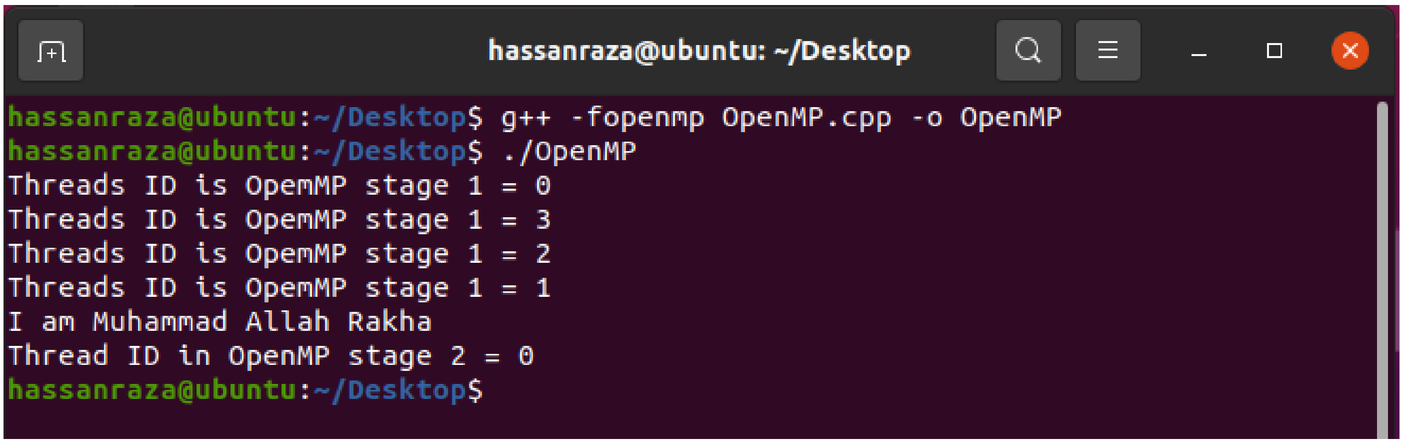
```
hassanraza@ubuntu:~/Desktop$ mpic++ mpi.cpp -o mpi
hassanraza@ubuntu:~/Desktop$ ls
core-site.xml  hdfs-site.xml    mpi      yarn-site.xml
hadoop-env.sh  mapred-site.xml  mpi.cpp
hassanraza@ubuntu:~/Desktop$ mpirun -np 4 mpi.out
--------------------------------------------------------------------------
mpirun was unable to find the specified executable file, and therefore
did not launch the job.  This error was first reported for process
rank 0; it may have occurred for other processes as well.

NOTE: A common cause for this error is misspelling a mpirun command
      line parameter option (remember that mpirun interprets the first
      unrecognized command line token as the executable).

Node:       ubuntu
Executable: mpi.out
--------------------------------------------------------------------------
4 total processes failed to start
hassanraza@ubuntu:~/Desktop$ mpirun -np 4 mpi
Muhammad Allah Rakha3
Muhammad Allah Rakha0
Muhammad Allah Rakha2
Muhammad Allah Rakha1
hassanraza@ubuntu:~/Desktop$
```

Open MP



```
hassanraza@ubuntu:~/Desktop$ gedit OpenMP.cpp
```

```
#include <iostream>
#include <omp.h>

using namespace std;

int main(int argc, char** argv){

 #pragma omp parallel num_threads(4)
 // create 4 threads and region inside it will be executed by all threads.
 {
   #pragma omp critical
   // allow one thread at a time to access below statement
   cout<<"Threads ID is OpemMP stage 1 = "<<omp_get_thread_num()<<endl;
 } // here all thread get merged into one thread id

 cout<<"I am Muhammad Allah Rakha"<<endl;

 #pragma omp parralel num_threads(2)
 // create 2 threads
 {
   cout<<"Thread ID in OpenMP stage 2 = "<<omp_get_thread_num()<<endl;
 }

}
```



```
hassanraza@ubuntu:~/Desktop$ g++ -fopenmp OpenMP.cpp -o OpenMP
hassanraza@ubuntu:~/Desktop$ ./OpenMP
Threads ID is OpemMP stage 1 = 0
Threads ID is OpemMP stage 1 = 3
Threads ID is OpemMP stage 1 = 2
Threads ID is OpemMP stage 1 = 1
I am Muhammad Allah Rakha
Thread ID in OpenMP stage 2 = 0
hassanraza@ubuntu:~/Desktop$
```