Name : Ansari M.Saeem M.Saleem

Uid : 2019430001

Subject : NAD

Expt no : 2

Aim : Programming with Open MPI.

Aim:

Programming with Open MPI.

Objectives:

- Establish a simple and limited set of directives for programming shared memory machines.
- To provide capability to incrementally parallelize a serial program, unlike message-passing libraries which typically require an all or nothing approach

Theory:

OpenMP -

Open Multi-processing (OpenMP) is a technique of parallelizing a section of C/C++/Fortran code. OpenMP is also seen as an extension to C/C++/Fortran languages by adding the parallelizing features to them. In general, OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms that range from the normal desktop computer to the high-end supercomputers.

Method for Creating Parallel Program:

Step 1. Include the header file: We have to include the OpenMP header for our program along with the standard header files.

```
//OpenMP header
#include <omp.h>
```

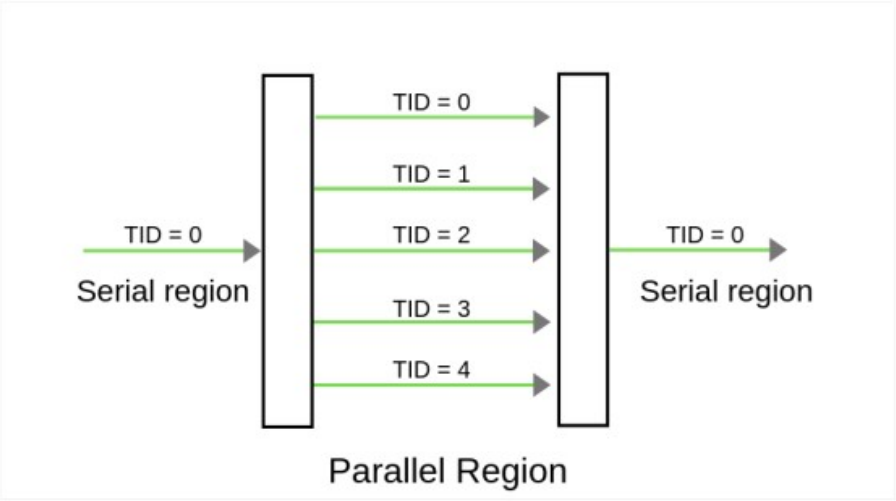
Step 2. Specify the parallel region: In OpenMP, we need to mention the region which we are going to make it as parallel using the keyword pragma omp parallel. The pragma omp parallel is used to fork additional threads to carry out the work enclosed in the parallel. The original thread will be denoted as the master thread with thread ID 0. Code for creating a parallel region would be,

```
#pragma omp parallel
{
        //Parallel region code
}
```

Step 3. Set the number of threads: we can set the number of threads to execute the program using the external variable.

```
export OMP_NUM_THREADS=5
```

As per the below figure, Once the compiler encounters the parallel regions code, the master thread(thread which has thread id 0) will fork into the specified number of threads. Here it will get forked into 5 threads because we will initialise the number of threads to be executed as 5, using the command export OMP_NUM_THREADS=5. Entire code within the parallel region will be executed by all threads concurrently. Once the parallel region ends, all threads will get merged into the master thread.



Program-

```
#include <stdio.h>
#include <omp.h>
int main()
{
#pragma omp parallel
{
        int ID=omp_get_thread_num();
        printf("Hello (%d)",ID);
        printf("World (%d)\n",ID);
}
}
```

CMD command to execute -

gcc -fopenmp openmp.c

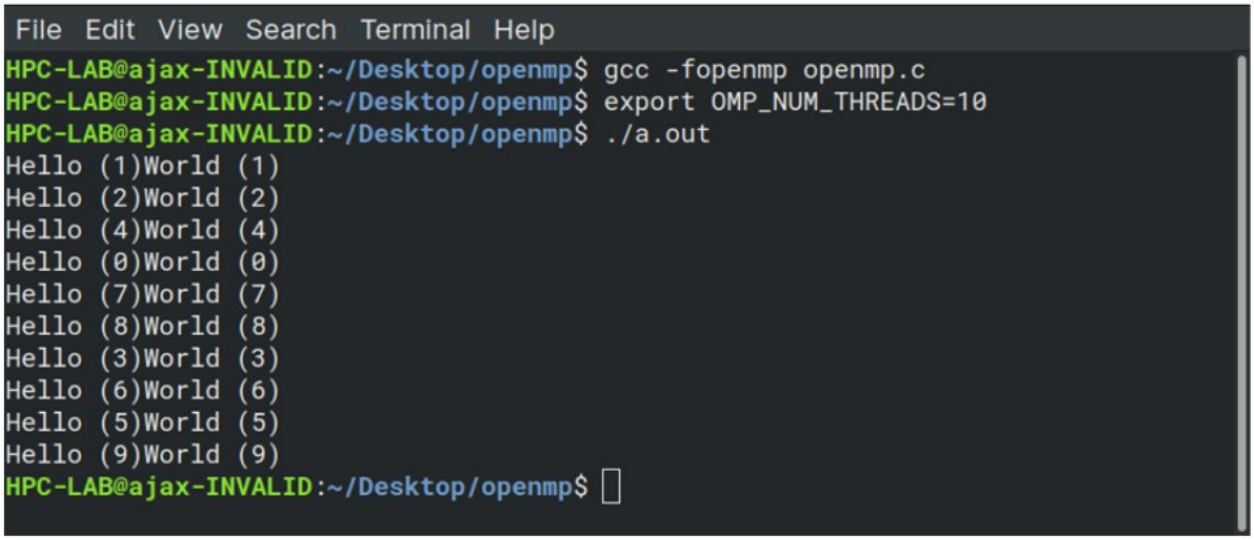
export OMP_NUM_THREADS=10

./a.out

Results:

1. Parallel print program using default thread count – 4

2. Parallel print program using thread count - 10



Conclusion:

We have successfully implemented an open MI program with different parameter and learn how to execute it.