Fall Sem 2021-22

Assignment: 1

Date : 16/08/21

Name : Ishan Sagar Jogalekar

Reg no : 19BCE2250

Course: Parallel and distributed computing LAB - CSE4001

Slot : L55+L56

Aim:

Write a simple OpenMP program to demonstrate the parallel loop construct.

a. Use OMP_SET_THREAD_NUM( ) and OMP_GET_THREAD_NUM( ) to find the number of processing unit

b. Use function invoke to print ‘Hello World’

c. To examine the above scenario, the functions such as omp_get_num_procs(), omp_set_num_threads(), omp_get_num_threads(), omp_in_parallel(), omp_get_dynamic() and omp_get_nested() are listed and the explanation is given below to explore the concept practically.

omp_set_num_threads() - takes an integer argument and requests that the Operating System provide that number of threads in subsequent parallel regions.

omp_get_num_threads() (integer function) - returns the actual number of threads in the current team of threads.

omp_get_thread_num() (integer function) - returns the ID of a thread, where the ID ranges from 0 to the number of threads minus 1. The thread with the ID of 0 is the master thread.

omp_get_num_procs() - returns the number of processors that are available when the function is called. omp_get_dynamic() - returns a value that indicates if the number of threads available in subsequent parallel region can be adjusted by the run time.

omp_get_nested( ) returns a value that indicates if nested parallelism is enabled.

SOURCE CODE:

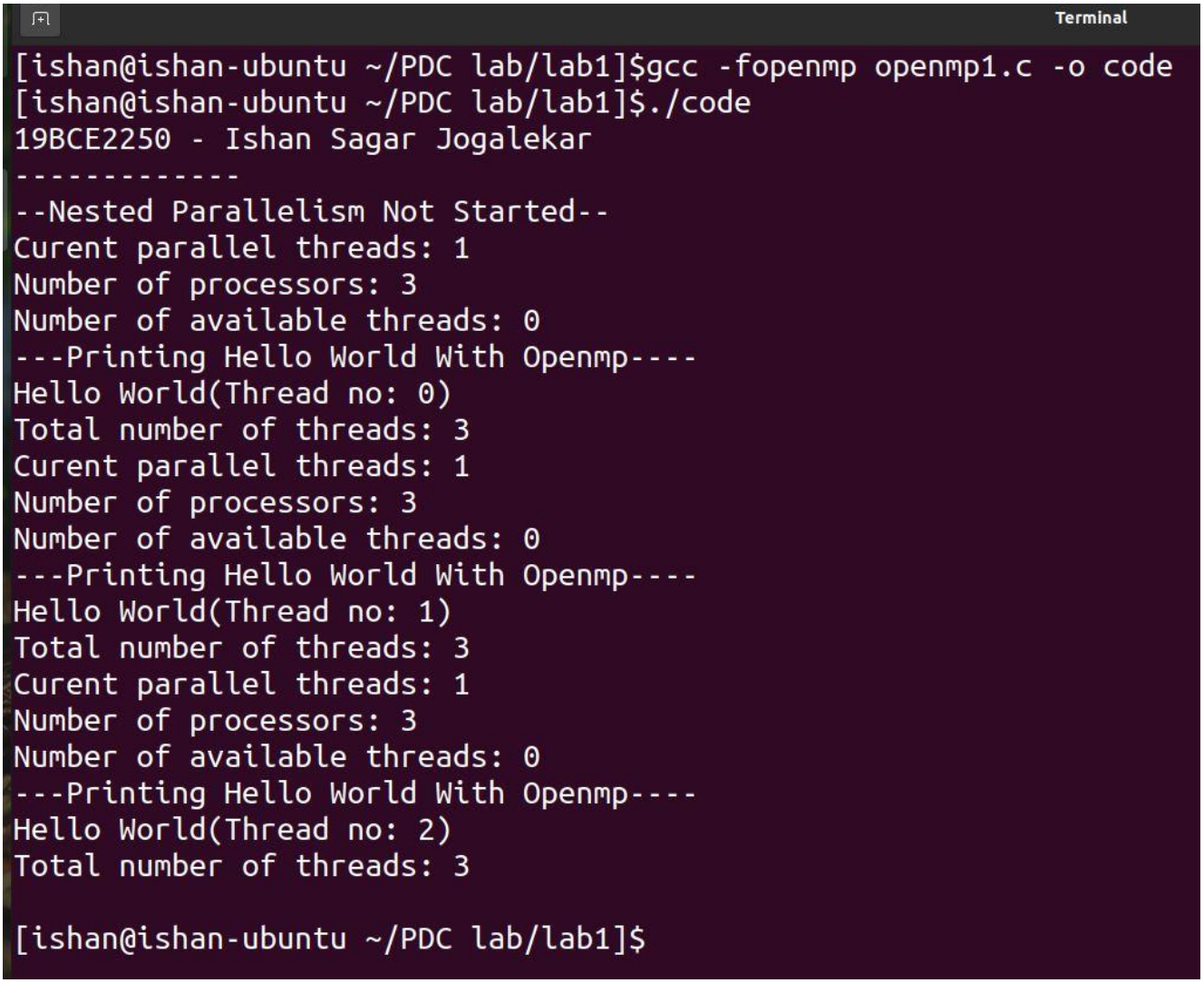
```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
void main(){
printf("19BCE2250 - Ishan Sagar Jogalekar\n");
int nest = omp_get_nested(); //Checking threads are nested or not
printf("-------------\n");
if(nest == 1){
printf("--Nested Parallelism Started--\n");
}
else{
printf("--Nested Parallelism Not Started--\n");
}
#pragma omp parallel
//starting paraalel code
{
int i = omp_in_parallel();
printf("Curent parallel threads: %d\n",i); //TO get curent true region of parallel threads

int t_ava = omp_get_dynamic(); //To get no of threads available

int num_p = omp_get_num_procs(); //number of available processors

omp_set_num_threads(6); //Requesting OS to set 6 threads
printf("Number of processors: %d\n",num_p);
printf("Number of available threads: %d\n",t_ava);
printf("---Printing Hello World With Openmp----\n");
int a = omp_get_thread_num(); //To get current number of running thread for program
int b = omp_get_num_threads();
printf("Hello World(Thread no: %d)\n",a);
printf("Total number of threads: %d\n",b);
}
printf("\n");
}
```

EXECUTION:

1.Program :

```
  GNU nano 4.8                                                                ope
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
void main(){
printf("19BCE2250 - Ishan Sagar Jogalekar\n");
int nest = omp_get_nested(); //Checking threads are nested or not
printf("-------------\n");
if(nest == 1){
printf("--Nested Parallelism Started--\n");
}
else{
printf("--Nested Parallelism Not Started--\n");
}
#pragma omp parallel
//starting paraalel code
{
int i = omp_in_parallel();
printf("Curent parallel threads: %d\n",i); //TO get curent true region of parallel threads

int t_ava = omp_get_dynamic(); //To get no of threads available

int num_p = omp_get_num_procs(); //number of available processors

omp_set_num_threads(6); //Requesting OS to set 6 threads

printf("Number of processors: %d\n",num_p);
printf("Number of available threads: %d\n",t_ava);
printf("---Printing Hello World With Openmp----\n");
int a = omp_get_thread_num(); //To get current number of running thread for program
int b = omp_get_num_threads();
printf("Hello World(Thread no: %d)\n",a);
printf("Total number of threads: %d\n",b);

}
printf("\n");
```

2.Output:



REMARKS:

The header file to be included for this experiment is “omp.h” which isOpenMP provided header file. It provides functions for parallel programming.

1.When checking for nested parallelism, we are getting return value of omp_get_nested returned is 0 hence using if-else statement output is nested parallelism is disabled.

2.#pragma open parallel container is used to contain the code which is to be parallelized

3.The number of processors printed out is 6 and hence the master process is forked into 6 parallelized sub-processes.

4.Using omp_get_thread_num() function we returned the actual number of threads (ID of thread)in the current team of threads and printed them infront of "Hello World" string.

5. omp_get_num_procs() function is used to returns the number of available processors. As per output current running processors are 3.

6. omp_get_num_threads() integer function is useful to return the actual number of threads in the current team of threads.For current scenario it is 3.

7. omp_get_dynamic() function returns a value that indicates if the number of threads available in subsequent parallel region can be adjusted by the run time.