NAME: MOUNVI PODAPATI

SLOT: L15 + L16

DATE: 16TH AUGUST 2021

FACULTY: PROF. DEEPAK B.D

ASSESSMENT NO. : 1

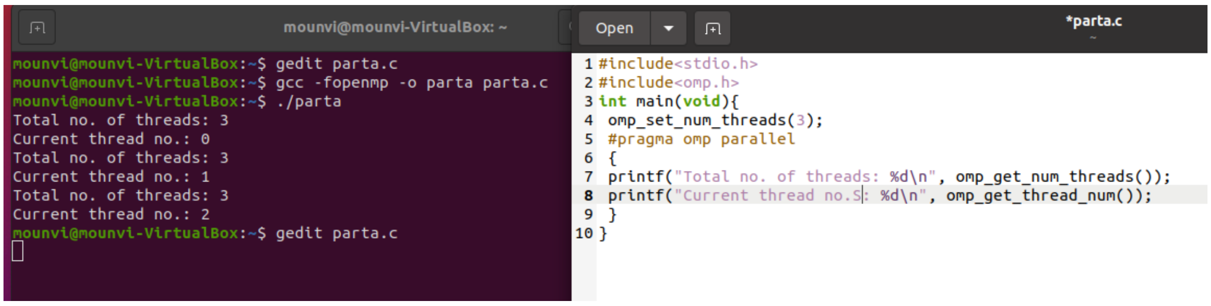
Aim: Write a simple OpenMP program to demonstrate the parallel loop construct.

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.

[a] SOURCE CODE:

```
#include<stdio.h>
#include<omp.h>
int main(void){
 omp_set_num_threads(3);
 #pragma omp parallel
 {
 printf("Total no. of threads: %d\n", omp_get_num_threads());
 printf("Current thread no.S: %d\n", omp_get_thread_num());
 }
}
```

EXECUTION:



```
mounvi@mounvi-VirtualBox:~$ gedit parta.c
mounvi@mounvi-VirtualBox:~$ gcc -fopenmp -o parta parta.c
mounvi@mounvi-VirtualBox:~$ ./parta
Total no. of threads: 3
Current thread no.: 0
Total no. of threads: 3
Current thread no.: 1
Total no. of threads: 3
Current thread no.: 2
mounvi@mounvi-VirtualBox:~$ gedit parta.c
```

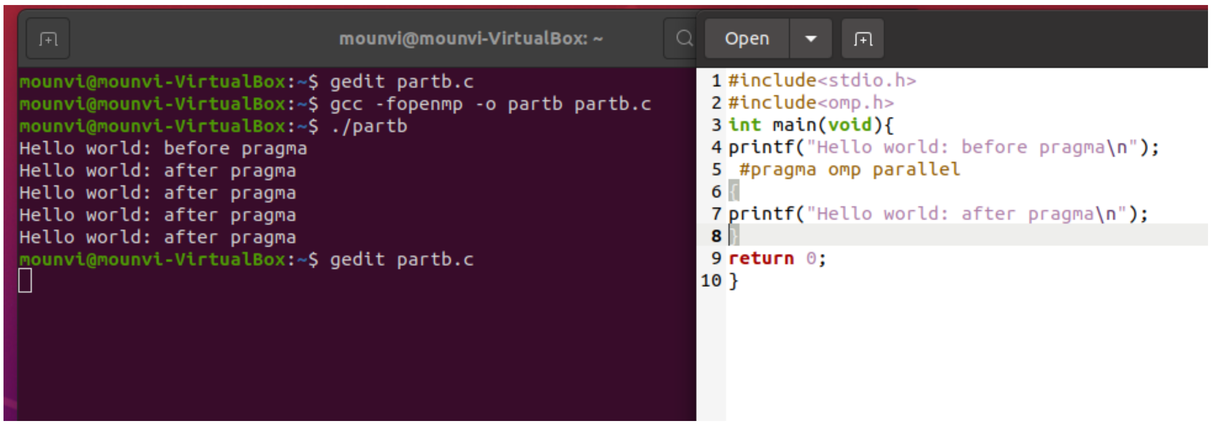
```
#include<stdio.h>
#include<omp.h>
int main(void){
 omp_set_num_threads(3);
 #pragma omp parallel
 {
 printf("Total no. of threads: %d\n", omp_get_num_threads());
 printf("Current thread no.S: %d\n", omp_get_thread_num());
 }
}
```

REMARKS:
Using omp_set_num_threads(int) Operating System is requested to provide 3 threads in subsequent parallel regions and this realized using omg_get_num_threads where it returns the actual number of threads in the current team of threads and the corresponding thread ID is found using omp_get_thread_num()

[b] SOURCE CODE:

```
#include<stdio.h>
#include<omp.h>
int main(void){
printf("Hello world: before pragma\n");
 #pragma omp parallel
{
printf("Hello world: after pragma\n");
}
return 0;
}
```

EXECUTION:



```
mounvi@mounvi-VirtualBox:~$ gedit partb.c
mounvi@mounvi-VirtualBox:~$ gcc -fopenmp -o partb partb.c
mounvi@mounvi-VirtualBox:~$ ./partb
Hello world: before pragma
Hello world: after pragma
Hello world: after pragma
Hello world: after pragma
Hello world: after pragma
mounvi@mounvi-VirtualBox:~$ gedit partb.c
```

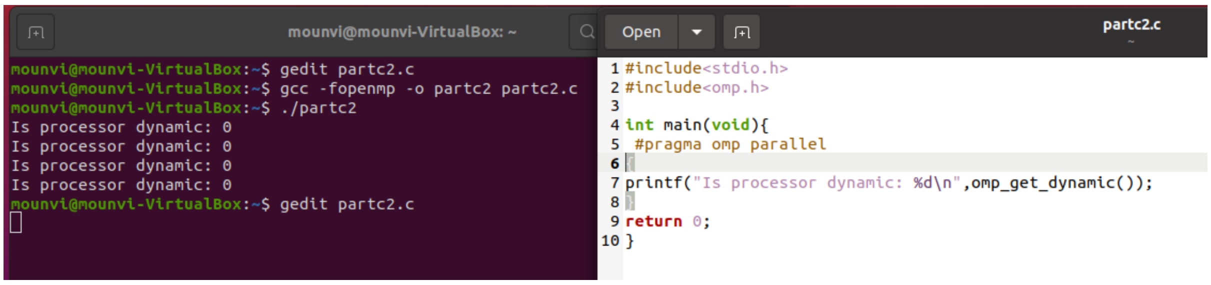
```
#include<stdio.h>
#include<omp.h>
int main(void){
printf("Hello world: before pragma\n");
 #pragma omp parallel
{
printf("Hello world: after pragma\n");
}
return 0;
}
```

REMARKS:
#pragma omp parallel is used to realize additional threads to carry out the work in parallel.

[c] 1. SOURCE CODE:

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

int main(void){
 #pragma omp parallel
 {
 printf("Is processor dynamic: %d\n", omp_get_dynamic());
 }
return 0;
}
```

EXECUTION:



```
mounvi@mounvi-VirtualBox:~$ gedit partc2.c
mounvi@mounvi-VirtualBox:~$ gcc -fopenmp -o partc2 partc2.c
mounvi@mounvi-VirtualBox:~$ ./partc2
Is processor dynamic: 0
Is processor dynamic: 0
Is processor dynamic: 0
Is processor dynamic: 0
mounvi@mounvi-VirtualBox:~$ gedit partc2.c
```

```c
#include<stdio.h>
#include<omp.h>

int main(void){
 #pragma omp parallel

printf("Is processor dynamic: %d\n",omp_get_dynamic());

return 0;
}
```

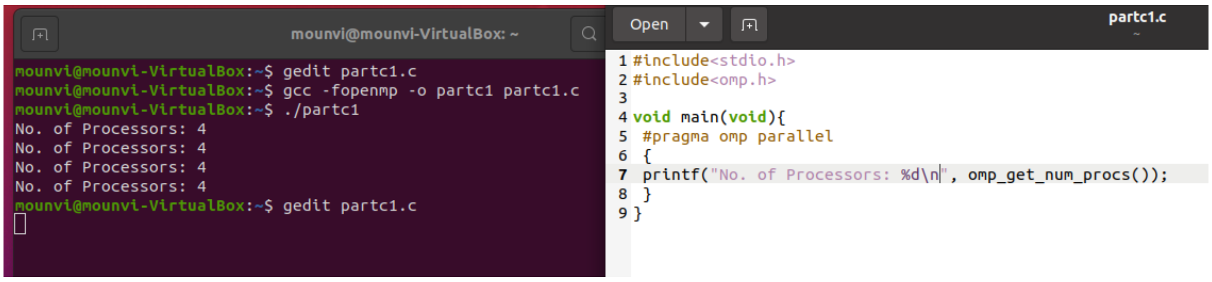
REMARKS:

Realized if the processor is dynamic using omp_get_dynamic() and this function returns 1 if the number of threads available in subsequent parallel region can be adjusted at run time else returns 0.

2. **SOURCE CODE:**

```c
#include<stdio.h>
#include<omp.h>

void main(void){
 #pragma omp parallel
 {
 printf("No. of Processors: %d\n", omp_get_num_procs());
 }
}
```

EXECUTION:



```
mounvi@mounvi-VirtualBox:~$ gedit partc1.c
mounvi@mounvi-VirtualBox:~$ gcc -fopenmp -o partc1 partc1.c
mounvi@mounvi-VirtualBox:~$ ./partc1
No. of Processors: 4
No. of Processors: 4
No. of Processors: 4
No. of Processors: 4
mounvi@mounvi-VirtualBox:~$ gedit partc1.c
```

```c
#include<stdio.h>
#include<omp.h>

void main(void){
 #pragma omp parallel
 {
 printf("No. of Processors: %d\n", omp_get_num_procs());
 }
}
```

REMARKS:

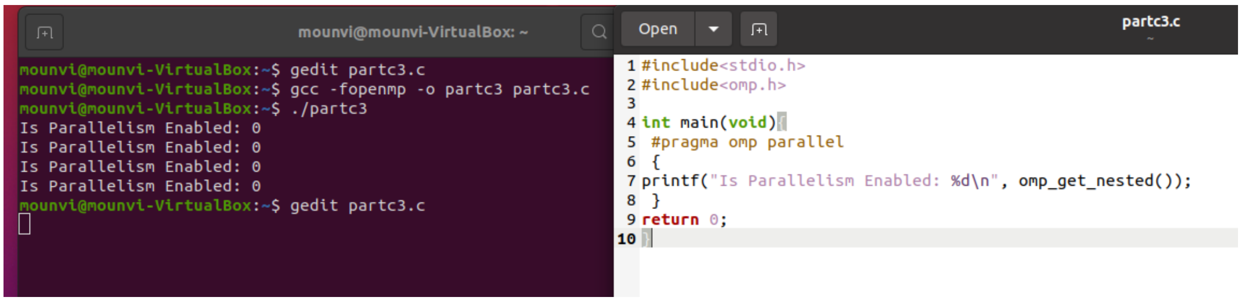
Number of processors available are returned by the function omp_get_num_procs().

3. **SOURCE CODE:**

```c
#include<stdio.h>

#include<omp.h>


int main(void){

 #pragma omp parallel

 {
```

```
printf("Is Parallelism Enabled: %d\n", omp_get_nested());
 }
return 0;
}
```

EXECUTION:



```
mounvi@mounvi-VirtualBox:~$ gedit partc3.c
mounvi@mounvi-VirtualBox:~$ gcc -fopenmp -o partc3 partc3.c
mounvi@mounvi-VirtualBox:~$ ./partc3
Is Parallelism Enabled: 0
Is Parallelism Enabled: 0
Is Parallelism Enabled: 0
Is Parallelism Enabled: 0
mounvi@mounvi-VirtualBox:~$ gedit partc3.c
```

```c
#include<stdio.h>
#include<omp.h>

int main(void){
 #pragma omp parallel
 {
printf("Is Parallelism Enabled: %d\n", omp_get_nested());
 }
return 0;
}
```

REMARKS:

The omp_get_nested() function returns ‘1’, if nested parallelism is enabled and ‘0’ if disabled.