NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL
DEPARTMENT OF INFORMATION TECHNOLOGY
IT 301 Parallel Computing LAB 2
3rd August 2021
Faculty: Dr. Geetha V

NAME: BHUVANESWAR DHARMASIVAM

ROLL NO:191IT107

1. Program 1 [2 Marks]

Aim: To understand and analyze shared clause in parallel directive.
Execute the program and write your observation. Change number of threads and write your observation.

```
/*shared.c*/
#include<omp.h>
int main()
{
int x=20;
#pragma omp parallel shared(x)
{int tid=omp_get_thread_num();
x=x+1;
printf("Thread [%d]\n value of x is %d",tid,x);}
}
```

No of threads=20,30.

```
bhuvan@bhuvan-N550JK:~/Desktop/IT301/Assignments/Assignment 2$ gcc -o shared -fopenmp shared.c
bhuvan@bhuvan-N550JK:~/Desktop/IT301/Assignments/Assignment 2$ ./shared
Thread [1]
 value of x is 21Thread [5]
 value of x is 21Thread [0]
 value of x is 21Thread [4]
 value of x is 22Thread [7]
 value of x is 23Thread [2]
 value of x is 21Thread [6]
 value of x is 21Thread [3]
bhuvan@bhuvan-N550JK:~/Desktop/IT301/Assignments/Assignment 2$ gcc -o shared -fopenmp shared.c
bhuvan@bhuvan-N550JK:~/Desktop/IT301/Assignments/Assignment 2$ ./shared
Thread [3]
 value of x is 31Thread [1]
 value of x is 33Thread [5]
 value of x is 33Thread [6]
 value of x is 33Thread [2]
 value of x is 33Thread [7]
 value of x is 34Thread [4]
 value of x is 32Thread [0]
```

Explanation: The variable 'x' is shared so the change in one threads visible in other threads.

2. Program 2 [2 Marks]

Learn the concept of private(), firstprivate()

(a) First execute the program with declaring i as *private(i).* Along with results , write your observation

```
bhuvan@bhuvan-N550JK:~/Desktop/IT301/Assignments/Assignment 2$ gcc -o learn -fopenmp learn.c
bhuvan@bhuvan-N550JK:~/Desktop/IT301/Assignments/Assignment 2$ ./learn
Value of i before pragma i=20
Value after entering pragma i=0 tid=0
Value after changing value i=0 tid=0
Value after entering pragma i=0 tid=2
Value after changing value i=2 tid=2
Value after entering pragma i=0 tid=1
Value after changing value i=1 tid=1
Value after entering pragma i=0 tid=3
Value after changing value i=3 tid=3
Value after having pragma i=20 tid=0
```

Explanation: The variable 'i' is private. It is 20 before parallel region but is not 20 after entering because it is private.

(b) Then execute the same program with *firstprivate(i).* Observe the results and write your observation.

```
bhuvan@bhuvan-N550JK:~/Desktop/IT301/Assignments/Assignment 2$ gcc -o learn -fopenmp learn.c
bhuvan@bhuvan-N550JK:~/Desktop/IT301/Assignments/Assignment 2$ ./learn
Value of i before pragma i=20
Value after entering pragma i=20 tid=0
Value after changing value i=20 tid=0
Value after entering pragma i=20 tid=1
Value after changing value i=21 tid=1
Value after entering pragma i=20 tid=2
Value after changing value i=22 tid=2
Value after entering pragma i=20 tid=3
Value after changing value i=23 tid=3
Value after having pragma i=20 tid=0
```

Explanation: 'i' is firstprivate so the value before parallel region is assigned to it in the amin thread.

```
/*learn.c*/
#include<stdio.h>
#include<omp.h>
int main()
{
int i=20;
printf("Value of i before pragma i=%d\n",i);
#pragma omp parallel num_threads(4) private(i)
{
printf("Value after entering pragma i=%d tid=%d\n",i, omp_get_thread_num());
i=i+omp_get_thread_num(); //adds thread_id to i
printf("Value after changing value i=%d tid=%d\n",i, omp_get_thread_num());
}
printf("Value after having pragma i=%d tid=%d\n",i, omp_get_thread_num());}
```

3. Programming exercise [6 Marks]

Write a parallel program to perform c[i]=a[i]+b[i] where i=0,1,2……N. Execute the program by varying number of elements and number of threads. Check the computation done by each thread.

CODE:

C qs3.c

home > bhuvan > Desktop > IT301 > Assignments > Assignment 2 > C qs3.c > main()

```c
#include<stdio.h>
#include<omp.h>
int main()
{
    int i,N=5;
    int a[5],b[5],c[N];

    //Initializing the array
    for(i=0;i<5;i++)
    {
        a[i]=i;
        b[i]=i;
    }

    //compute parallel
    #pragma omp parallel
        #pragma omp for
        for(i=0;i<5;i++)
        {
            int tid=omp_get_thread_num();
            c[i]=a[i]+b[i];
            printf("Computation at Thread [%d]=%d\n",tid,c[i]);
        }
}
```

EXECUTION:

```
bhuvan@bhuvan-N550JK:~/Desktop/IT301/Assignments/Assignment 2$ gcc -o qs -fopenmp qs3.c
bhuvan@bhuvan-N550JK:~/Desktop/IT301/Assignments/Assignment 2$ ./qs
Computation at Thread [0]=0
Computation at Thread [4]=8
Computation at Thread [1]=2
Computation at Thread [3]=6
Computation at Thread [2]=4
```

Explanation: The loop construct iterates and executes computation done in each thread.