NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL
DEPARTMENT OF INFORMATION TECHNOLOGY

IT 301 Parallel Computing LAB 3

1st September 2020

Faculty: Dr. Geetha V and Mrs. Tanmayee

Name: Chinmayi C. Ramakrishna
Roll No.: 181IT113

Program 1:
Execute following code and observe the working of threadprivate directive and copyin clause:

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program1 -fopenmp Program1.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program1
Parallel Region 1
Thread  1 Value of x is 12
Thread  2 Value of x is 10
Thread  3 Value of x is 10
Thread  0 Value of x is 11
Parallel Region 2
Thread 3 Value of x is 10
Thread 1 Value of x is 12
Thread 2 Value of x is 10
Thread 0 Value of x is 11
Value of x in Main Region is 11
PS C:\Users\Chinmayi\Cpp Codes> ./Program1
```

With copyin() clause. The master thread value is copied to the rest of the threads. These threads increment the initial x=10.

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program1 -fopenmp Program1.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program1
Parallel Region 1
Thread  1 Value of x is 2
Thread  3 Value of x is 0
Thread  2 Value of x is 0
Thread  0 Value of x is 11
Parallel Region 2
Thread 1 Value of x is 2
Thread 2 Value of x is 0
Thread 3 Value of x is 0
Thread 0 Value of x is 11
Value of x in Main Region is 11
PS C:\Users\Chinmayi\Cpp Codes>
```

Without using copyin () clause. The master thread value is not incremented by other threads. Hence, the value of thread 1 is not 10+2 =12. Thread 0 has value 10 and rest of the threads (thread 1, thread 2, thread 3) have x value as 0.

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program1 -fopenmp Program1.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program1
Parallel Region 1
Thread  1 Value of x is 12
Thread  2 Value of x is 10
Thread  3 Value of x is 10
Thread  0 Value of x is 11
Parallel Region 2
Thread 1 Value of x is 12
Thread 2 Value of x is 10
Thread 3 Value of x is 10
Thread 0 Value of x is 11
Value of x in Main Region is 11
```

Without using copyin () clause and initializing x globally. The output is same as using copyin() clause. By initializing x globally, all the threads have access to the same value.

Program 2:

Learn the concept of firstprivate () and threadprivate ().

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program2 -fopenmp Program2.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program2
1. count=0
tid=0,a[0]=0, count=1 x=11
tid=0,a[1]=2, count=2 x=12
tid=0,a[2]=4, count=3 x=13
tid=0,a[3]=6, count=4 x=14
tid=0,a[4]=8, count=5 x=15
tid=1,a[5]=10, count=1 x=11
tid=1,a[6]=12, count=2 x=12
tid=1,a[7]=14, count=3 x=13
tid=1,a[8]=16, count=4 x=14
tid=1,a[9]=18, count=5 x=15
2. before copyprivate count=5 x=10 tid=1
2. before copyprivate count=5 x=10 tid=0
3. after copyprivate count=25 x=10 tid=0
tid=0,a[0]=0, count=26, x=11
tid=0,a[1]=1, count=27, x=12
tid=0,a[2]=4, count=28, x=13
tid=0,a[3]=9, count=29, x=14
tid=0,a[4]=16, count=30, x=15
3. after copyprivate count=25 x=10 tid=1
tid=1,a[5]=25, count=26, x=11
tid=1,a[6]=36, count=27, x=12
tid=1,a[7]=49, count=28, x=13
tid=1,a[8]=64, count=29, x=14
tid=1,a[9]=81, count=30, x=15
4. count=30 x=10
```

threadprivate variables are able to persist between parallel sections of the code. x value remains the same for both the threads and hence their values remain same.

firstprivate() ensures that every thread has its own instance of the variable.

Program 3:

Program to understand the concept of collapse ().

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program3 -fopenmp Program3.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program3
tid=1, i=3  j=0
tid=1, i=3  j=1
tid=1, i=3  j=2
tid=1, i=3  j=3
tid=1, i=3  j=4
tid=1, i=4  j=0
tid=1, i=4  j=1
tid=1, i=4  j=2
tid=1, i=4  j=3
tid=1, i=4  j=4
tid=1, i=5  j=0
tid=0, i=0  j=0
tid=0, i=0  j=1
tid=0, i=0  j=2
tid=0, i=0  j=3
tid=1, i=5  j=1
tid=0, i=0  j=4
tid=0, i=1  j=0
tid=0, i=1  j=1
tid=0, i=1  j=2
tid=1, i=5  j=2
tid=1, i=5  j=3
tid=1, i=5  j=4
tid=0, i=1  j=3
tid=0, i=1  j=4
tid=0, i=2  j=0
tid=0, i=2  j=1
tid=0, i=2  j=2
tid=0, i=2  j=3
tid=0, i=2  j=4
PS C:\Users\Chinmayi\Cpp Codes>
```

With two for loops.

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program3 -fopenmp Program3.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program3
tid=0, i=0  j=0  k=0
tid=0, i=0  j=0  k=1
tid=0, i=0  j=1  k=0
tid=0, i=0  j=1  k=1
tid=0, i=1  j=0  k=0
tid=0, i=1  j=0  k=1
tid=0, i=1  j=1  k=0
tid=0, i=1  j=1  k=1
tid=0, i=2  j=0  k=0
tid=0, i=2  j=0  k=1
tid=0, i=2  j=1  k=0
tid=0, i=2  j=1  k=1
PS C:\Users\Chinmayi\Cpp Codes>
```

Three for loops and collapse () clause. A single thread performs all the iterations.

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program3 -fopenmp Program3.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program3
tid=0, i=0  j=0  k=0
tid=1, i=1  j=1  k=0
tid=1, i=1  j=1  k=1
tid=1, i=2  j=0  k=0
tid=1, i=2  j=0  k=1
tid=1, i=2  j=1  k=0
tid=1, i=2  j=1  k=1
tid=0, i=0  j=0  k=1
tid=0, i=0  j=1  k=0
tid=0, i=0  j=1  k=1
tid=0, i=1  j=0  k=0
tid=0, i=1  j=0  k=1
PS C:\Users\Chinmayi\Cpp Codes>
```

Three for loops and collapse (2). It collapses two for loops into one large iteration space and divides it among two threads.

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program3 -fopenmp Program3.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program3
tid=3, i=2  j=0  k=1
tid=3, i=2  j=1  k=0
tid=3, i=2  j=1  k=1
tid=0, i=0  j=0  k=0
tid=0, i=0  j=0  k=1
tid=0, i=0  j=1  k=0
tid=1, i=0  j=1  k=1
tid=1, i=1  j=0  k=0
tid=1, i=1  j=0  k=1
tid=2, i=1  j=1  k=0
tid=2, i=1  j=1  k=1
tid=2, i=2  j=0  k=0
PS C:\Users\Chinmayi\Cpp Codes>
```

Three for loops and collapse (3). It collapses three for loops into one large iteration space and divides it among four threads.

The functionality of collapse it to collapse the for loops into a single iteration. Increasing the collapse value increases parallelism as it assigns more number of threads.

Program 4:

How to compare sequential and parallel program execution times.?

Program4.c > main(void)

```c
#include <stdio.h>
#include <sys/time.h>
#include <omp.h>
#include <stdlib.h>
#define n 100000
int main(void)
{
    struct timeval TimeValue_Start;
    struct timezone TimeZone_Start;
    struct timeval TimeValue_Final;
    struct timezone TimeZone_Final;
    long time_start, time_end;
    double time_overhead;double pi,x;
    int i;
    int array[n];
    for(i = 0; i<=n; i++)
    {
        array[i] = rand();
    }

    gettimeofday(&TimeValue_Start, &TimeZone_Start);
    int small = INT_MAX;
    #pragma omp parallel for schedule(runtime) reduction(min:small)
        for(i = 0; i<=n; i++)
        {
            if(array[i]<small) small = array[i];
        }

    gettimeofday(&TimeValue_Final, &TimeZone_Final);
    time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
    time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
    time_overhead = (time_end - time_start)/1000000.0;
    printf("\n\nTime in Seconds (T) : %lf\n",time_overhead);
    printf("\n\nSmallest element of the array is : %d\n",small);
    return 0;
}

```

Code for finding execution time of parallel execution for finding the smallest element in an array.

Program4.c > main(void)

```c
#include <stdio.h>
#include <sys/time.h>
#include <omp.h>
#include <stdlib.h>
#define n 100000
int main(void)
{
    struct timeval TimeValue_Start;
    struct timezone TimeZone_Start;
    struct timeval TimeValue_Final;
    struct timezone TimeZone_Final;
    long time_start, time_end;
    double time_overhead;double pi,x;
    int i;
    int array[n];

    for(i = 0; i<=n; i++)
    {
        array[i] = rand();
    }

    gettimeofday(&TimeValue_Start, &TimeZone_Start);
    int small = array;

        for(i = 0; i<=n; i++)
        {
            if(array[i] <= small) small = array[i];
        }

    gettimeofday(&TimeValue_Final, &TimeZone_Final);
    time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
    time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
    time_overhead = (time_end - time_start)/1000000.0;
    printf("\n\nTime in Seconds (T) : %lf\n",time_overhead);
    printf("\n\nSmallest element of the array is : %d\n",small);
    return 0;
}
```

Code for finding execution time of sequential execution for finding the smallest element in an array.

Outputs:

Using sequential execution.

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 5000


Time in Seconds (T) : 000011
Smallest element of the array is : 3
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 10000


Time in Seconds (T) : 000021


Smallest element of the array is : 19
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 50000


Time in Seconds (T) : 000097


Smallest element of the array is : 4
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 100000


Time in Seconds (T) : 000192


Smallest element of the array is : 13
PS C:\Users\Chinmayi\Cpp Codes>
```

Using schedule(static)

```
n = 5000


Time in Seconds (T) : 0.003059


Smallest element of the array is : 8
n = 10000


Time in Seconds (T) : 0.001825


Smallest element of the array is : 3
n = 50000


Time in Seconds (T) : 0.000996


Smallest element of the array is : 0
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 100000


Time in Seconds (T) : 0.000992


Smallest element of the array is : 0
```

Using schedule(static,1)

```
n = 5000


Time in Seconds (T) : 0.001444


Smallest element of the array is : 8
n = 10000


Time in Seconds (T) : 0.002030


Smallest element of the array is : 3
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 50000


Time in Seconds (T) : 0.002039


Smallest element of the array is : 0
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 100000


Time in Seconds (T) : 0.001960


Smallest element of the array is : 0
PS C:\Users\Chinmayi\Cpp Codes>
```

Using schedule(dynamic, 1)

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4

Time in Seconds (T) : 0.000997


Smallest element of the array is : 8
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4

Time in Seconds (T) : 0.002621


Smallest element of the array is : 3
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4

Time in Seconds (T) : 0.002916


Smallest element of the array is : 0
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 100000


Time in Seconds (T) : 0.004300


Smallest element of the array is : 0
PS C:\Users\Chinmayi\Cpp Codes>
```

Using schedule(guided)

```
n = 5000


Time in Seconds (T) : 0.001918


Smallest element of the array is : 8
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 10000


Time in Seconds (T) : 0.003052


Smallest element of the array is : 3
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 50000


Time in Seconds (T) : 0.002709


Smallest element of the array is : 0
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 100000


Time in Seconds (T) : 0.001004


Smallest element of the array is : 0
PS C:\Users\Chinmayi\Cpp Codes>
```

Using schedule(runtime)

```
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 5000


Time in Seconds (T) : 0.002010


Smallest element of the array is : 8
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 10000


Time in Seconds (T) : 0.001105


Smallest element of the array is : 3
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 50000


Time in Seconds (T) : 0.002913


Smallest element of the array is : 0
PS C:\Users\Chinmayi\Cpp Codes> gcc -o Program4 -fopenmp Program4.c
PS C:\Users\Chinmayi\Cpp Codes> ./Program4
n = 100000


Time in Seconds (T) : 0.004941


Smallest element of the array is : 0
PS C:\Users\Chinmayi\Cpp Codes>
```

At array size = 1M the parallel execution time is less than sequential execution time.

Table:

Schedule()	Total Execution time for number of iterations 5K	Total execution for number of iterations 10K	Total execution for number of iterations 50K	Total execution for number of iterations 100K
Sequential execution	0.000011 s	0.000021 s	0.000097 s	0.000192 s
static	0.003059 s	0.001825 s	0.000996 s	0.000992 s
Static, chunksize	0.001444 s	0.002030 s	0.002039 s	0.001960 s
Dynamic, chunksize	0.000997 s	0.002621 s	0.002916 s	0.004300 s
Guided	0.001918 s	0.003052 s	0.002709 s	0.001004 s
runtime	0.002010 s	0.001105 s	0.002913 s	0.004941 s