

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL
DEPARTMENT OF INFORMATION TECHNOLOGY
IT 301 Parallel Computing LAB 2

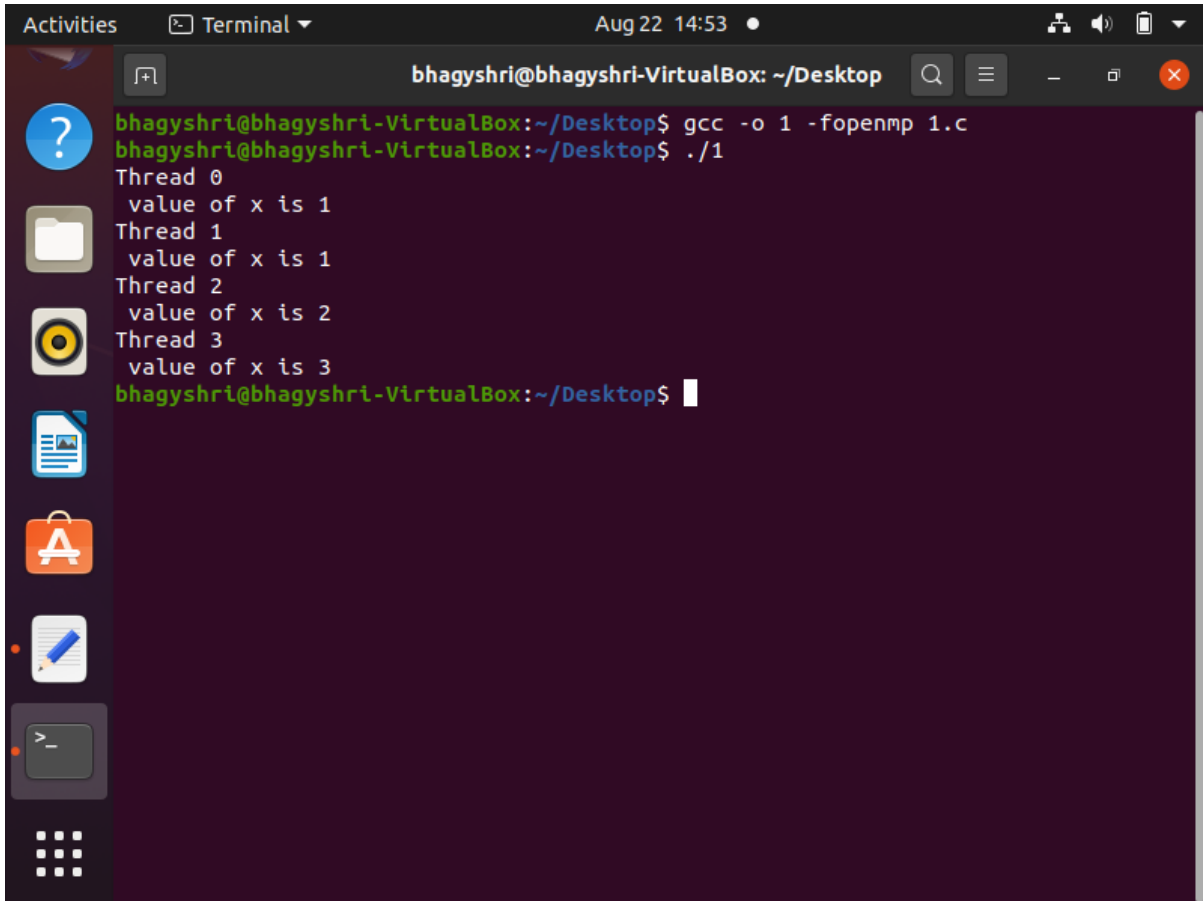
19th August 2020

Faculty: Dr. Geetha V and Mrs. Tanmayee

Bhagyashri Bhamare 181IT111

Program 1:

To understand and analyze shared clause in parallel directive.



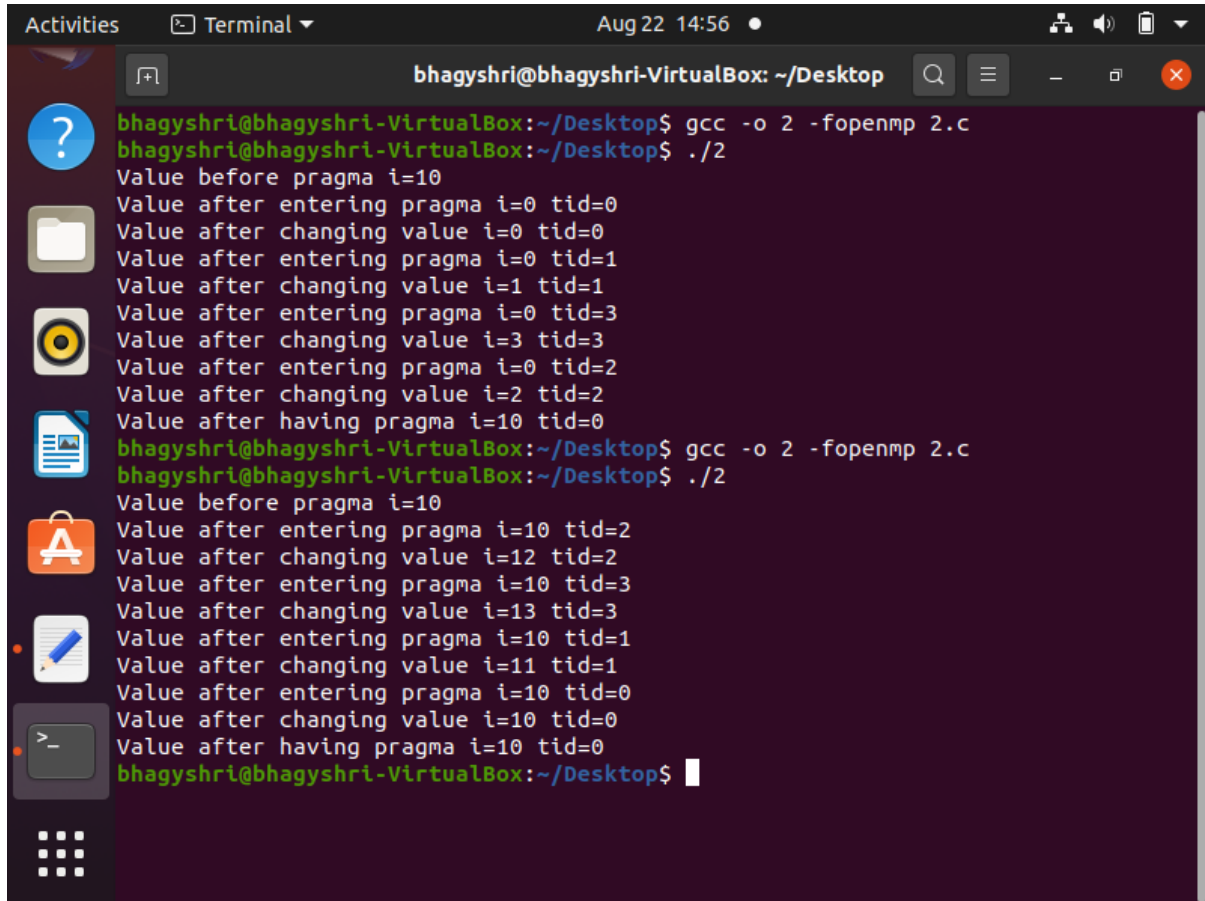
```
Activities Terminal Aug 22 14:53
bhagyshri@bhagyshri-VirtualBox: ~/Desktop
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ gcc -o 1 -fopenmp 1.c
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ ./1
Thread 0
  value of x is 1
Thread 1
  value of x is 1
Thread 2
  value of x is 2
Thread 3
  value of x is 3
bhagyshri@bhagyshri-VirtualBox:~/Desktop$
```

Value 1 is shared by three threads 1,0

2. Program 2

Learn the concept of `private ()`, `firstprivate ()`

private() Each thread 0, 1, 2 and 3 has its own instance of variable `i=0`



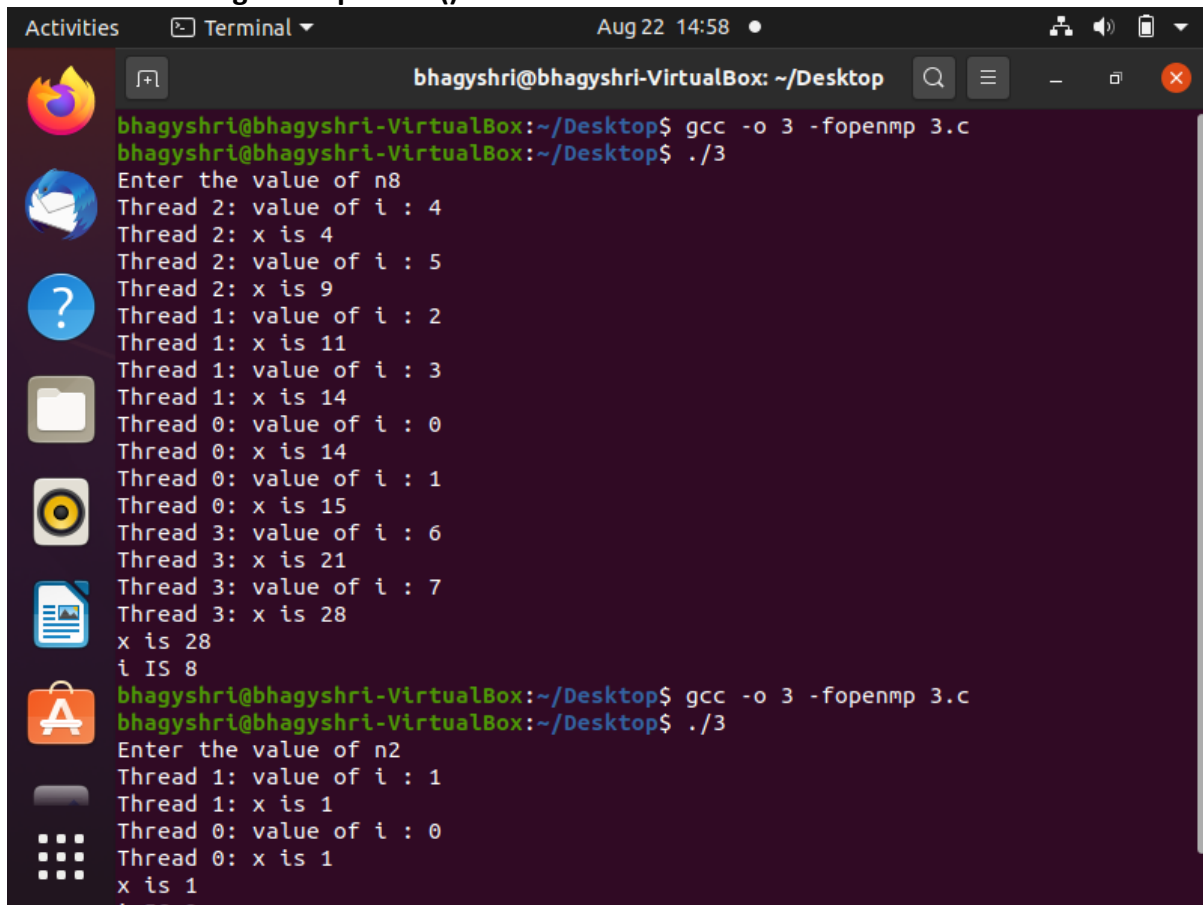
```
Aug 22 14:56 •
bhagyshri@bhagyshri-VirtualBox: ~/Desktop
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ gcc -o 2 -fopenmp 2.c
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ ./2
Value before pragma i=10
Value after entering pragma i=0 tid=0
Value after changing value i=0 tid=0
Value after entering pragma i=0 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 entering pragma i=0 tid=2
Value after changing value i=2 tid=2
Value after having pragma i=10 tid=0
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ gcc -o 2 -fopenmp 2.c
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ ./2
Value before pragma i=10
Value after entering pragma i=10 tid=2
Value after changing value i=12 tid=2
Value after entering pragma i=10 tid=3
Value after changing value i=13 tid=3
Value after entering pragma i=10 tid=1
Value after changing value i=11 tid=1
Value after entering pragma i=10 tid=0
Value after changing value i=10 tid=0
Value after having pragma i=10 tid=0
bhagyshri@bhagyshri-VirtualBox:~/Desktop$
```

firstprivate()

Every thread 0, 1, 2 and 3 has its own instance of the variable and the variable is initialized with the value of the variable. The threads 0, 1, 2 and 3 have `i=10`.

3. Program 3

Learn the working of lastprivate () clause:



```
Activities Terminal Aug 22 14:58
bhagyshri@bhagyshri-VirtualBox: ~/Desktop
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ gcc -o 3 -fopenmp 3.c
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ ./3
Enter the value of n8
Thread 2: value of i : 4
Thread 2: x is 4
Thread 2: value of i : 5
Thread 2: x is 9
Thread 1: value of i : 2
Thread 1: x is 11
Thread 1: value of i : 3
Thread 1: x is 14
Thread 0: value of i : 0
Thread 0: x is 14
Thread 0: value of i : 1
Thread 0: x is 15
Thread 3: value of i : 6
Thread 3: x is 21
Thread 3: value of i : 7
Thread 3: x is 28
x is 28
i IS 8
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ gcc -o 3 -fopenmp 3.c
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ ./3
Enter the value of n2
Thread 1: value of i : 1
Thread 1: x is 1
Thread 0: value of i : 0
Thread 0: x is 1
x is 1
i IS 0
```

The value of x at the next iteration is i of the current iteration+ value of x at the previous iteration. The variable that is set equal to the private version of a particular thread executes the final iteration or the last section.

4. Program 4

Demonstration of reduction clause in parallel directive.

```
Activities  Terminal  Aug 22 15:11  bhagyshri@bhagyshri-VirtualBox: ~/Desktop

bhagyshri@bhagyshri-VirtualBox:~/Desktop$ gcc -o 4 -fopenmp 4.c
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ ./4
Hi from 1
  value of x : 1
Hi from 4
  value of x : 1
Hi from 5
  value of x : 1
Hi from 2
  value of x : 1
Hi from 3
  value of x : 1
Hi from 0
  value of x : 1
Final x:6
bhagyshri@bhagyshri-VirtualBox:~/Desktop$
```

Question 1) 1. Write a parallel program to calculate the sum of elements in an array.

```
#include<stdio.h>
#include<omp.h>
#include<stdlib.h>
int main(int argc , char **argv)
{
    double *Array, sum;
    int array_size, i, threads, number;
    if( argc != 3 )
    {
        printf("Very Few Arguments\n ");
        printf("Syntax : exec <Threads> <array-size>\n");
        exit(-1);
    }

    threads=atoi(argv[1]);
    if ((threads!=1) && (threads!=2) && (threads!=4) &&
the execution of program. \n\n");
        printf("\nNumber of threads should be 1,2,4,8 or 16 for
        exit(-1);
    }
    array_size=atoi(argv[2]);
    if (array_size <= 0) {
        printf("\nArray Size Should Be Of Positive Value ");
        exit(1);
    }

    printf("\nThreads      : %d ",threads);
    printf("\nArray Size   : %d ",array_size);
    Array = (double *) malloc(sizeof(double) * array_size);
    printf("\nArray elements: ");

    for (i = 0; i < array_size; i++) {
        scanf("%d",&number);
        Array[i] = number;
    }

    sum=0.0;
    omp_set_num_threads(threads);
#pragma omp parallel for
        for (i = 0; i < array_size; i++)
    {
#pragma omp critical
        sum = sum + Array[i];
    }

    free(Array);

    printf("\nThe Sum Of Elements Of The Array Is: %lf\n", sum);
    return 0;
}
```

Activities Terminal Aug 22 15:19

bhagyshri@bhagyshri-VirtualBox: ~/Desktop

```
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ gcc -o sum -fopenmp sum.c
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ ./sum 4 5

Threads      : 4
Array Size   : 5
Array elements: 4 5 6 9 7

The Sum Of Elements Of The Array Is: 31.000000
bhagyshri@bhagyshri-VirtualBox:~/Desktop$
```

2. Write a parallel program to calculate the $a[i]=b[i]+c[i]$, for all elements in array $b[]$ and $c[]$.

```
#include<stdio.h>
#include<omp.h>
#include<stdlib.h>
int main(int argc , char **argv)
{
    double *Array, *Array1, *Array2, sum;
    int array_size, i, threads, number;
    if( argc != 3 )
    {
        printf("Very Few Arguments\n ");
        printf("Syntax : exec <Threads> <array-size>\n");
        exit(-1);
    }

    threads=atoi(argv[1]);
    if ((threads!=1) && (threads!=2) && (threads!=4) &&
    (threads!=8) && (threads!= 16) )
    {
        printf("\nNumber of threads should be 1,2,4,8 or 16 for
the execution of program. \n\n");
        exit(-1);
    }
    array_size=atoi(argv[2]);
    if (array_size <= 0) {
        printf("\nArray Size Should Be Of Positive Value ");
        exit(1);
    }

    printf("\nThreads      : %d ",threads);
    printf("\nArray Size   : %d ",array_size);
    Array = (double *) malloc(sizeof(double) * array_size);
    Array1 = (double *) malloc(sizeof(double) * array_size);
    Array2 = (double *) malloc(sizeof(double) * array_size);
    printf("\nArray1 elements: ");

    for (i = 0; i < array_size; i++) {
        scanf("%d",&number);
        Array1[i] = number;
    }
    printf("\nArray2 elements: ");

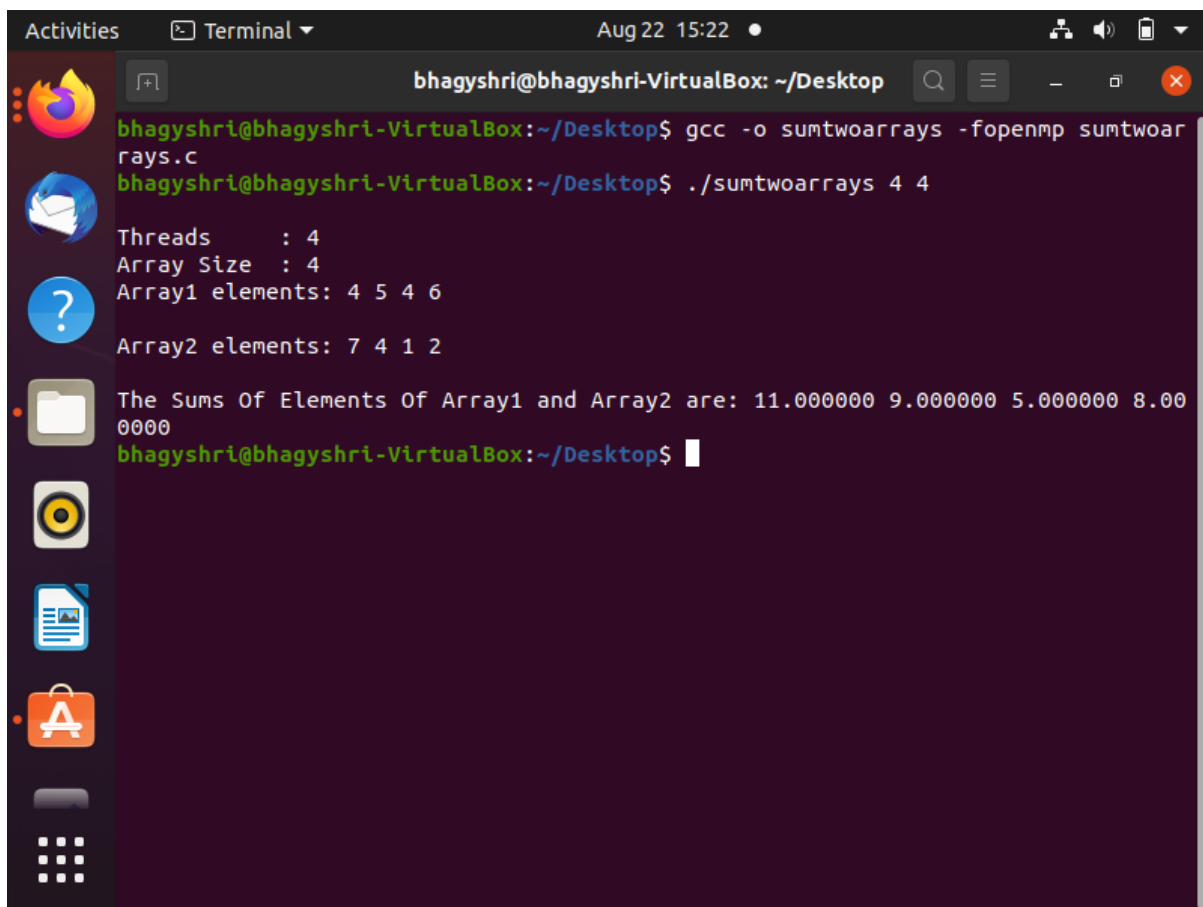
    for (i = 0; i < array_size; i++) {
        scanf("%d",&number);
        Array2[i] = number;
    }
    omp_set_num_threads(threads);
#pragma omp parallel for
    for (i = 0; i < array_size; i++)
    {
#pragma omp critical
        Array[i] = Array1[i] + Array2[i];
    }
}
```

```

printf("\nThe Sums Of Elements Of Array1 and Array2 are: ");
for (i = 0; i < array_size; i++)
{
    printf("%lf ",Array[i]);
}
printf("\n");
free(Array1);
free(Array2);
free(Array);
return 0;
}

```

output



```

Activities  Terminal  Aug 22 15:22
bhagyshri@bhagyshri-VirtualBox: ~/Desktop
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ gcc -o sumtwoarrays -fopenmp sumtwoarrays.c
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ ./sumtwoarrays 4 4
Threads      : 4
Array Size   : 4
Array1 elements: 4 5 4 6
Array2 elements: 7 4 1 2
The Sums Of Elements Of Array1 and Array2 are: 11.000000 9.000000 5.000000 8.000000
bhagyshri@bhagyshri-VirtualBox:~/Desktop$

```


3. Write a parallel program to find the largest among all elements in an array.

```
#include<stdio.h>
#include<omp.h>
#include<stdlib.h>
int main(int argc , char **argv)
{
    double *Array, largest;
    int array_size, i, threads, number;
    if( argc != 3 )
    {
        printf("Very Few Arguments\n ");
        printf("Syntax : exec <Threads> <array-size>\n");
        exit(-1);
    }

    threads=atoi(argv[1]);
    if ((threads!=1) && (threads!=2) && (threads!=4) &&
    (threads!=8) && (threads!= 16) )
    {
        printf("\nNumber of threads should be 1,2,4,8 or 16 for
the execution of program. \n\n");
        exit(-1);
    }
    array_size=atoi(argv[2]);
    if (array_size <= 0) {
        printf("\nArray Size Should Be Of Positive Value ");
        exit(1);
    }

    printf("\nThreads      : %d ",threads);
    printf("\nArray Size   : %d ",array_size);
    Array = (double *) malloc(sizeof(double) * array_size);
    printf("\nArray elements: ");

    for (i = 0; i < array_size; i++) {
        scanf("%d",&number);
        Array[i] = number;
    }
    largest=Array[0];
    omp_set_num_threads(threads);
#pragma omp parallel for
    for (i = 0; i < array_size; i++)
    {
#pragma omp critical
        if(Array[i]>=largest) largest = Array[i];
    }

    printf("\nThe Largest Element in the Array Is: %lf\n", largest);
    free(Array);
    return 0;
}
```

output

Activities Terminal Aug 22 15:25

bhagyshri@bhagyshri-VirtualBox: ~/Desktop

```
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ gcc -o largest -fopenmp largest.c
bhagyshri@bhagyshri-VirtualBox:~/Desktop$ ./largest 4 5

Threads      : 4
Array Size   : 5
Array elements: 4 5 4 6 8

The Largest Element in the Array Is: 8.000000
bhagyshri@bhagyshri-VirtualBox:~/Desktop$
```