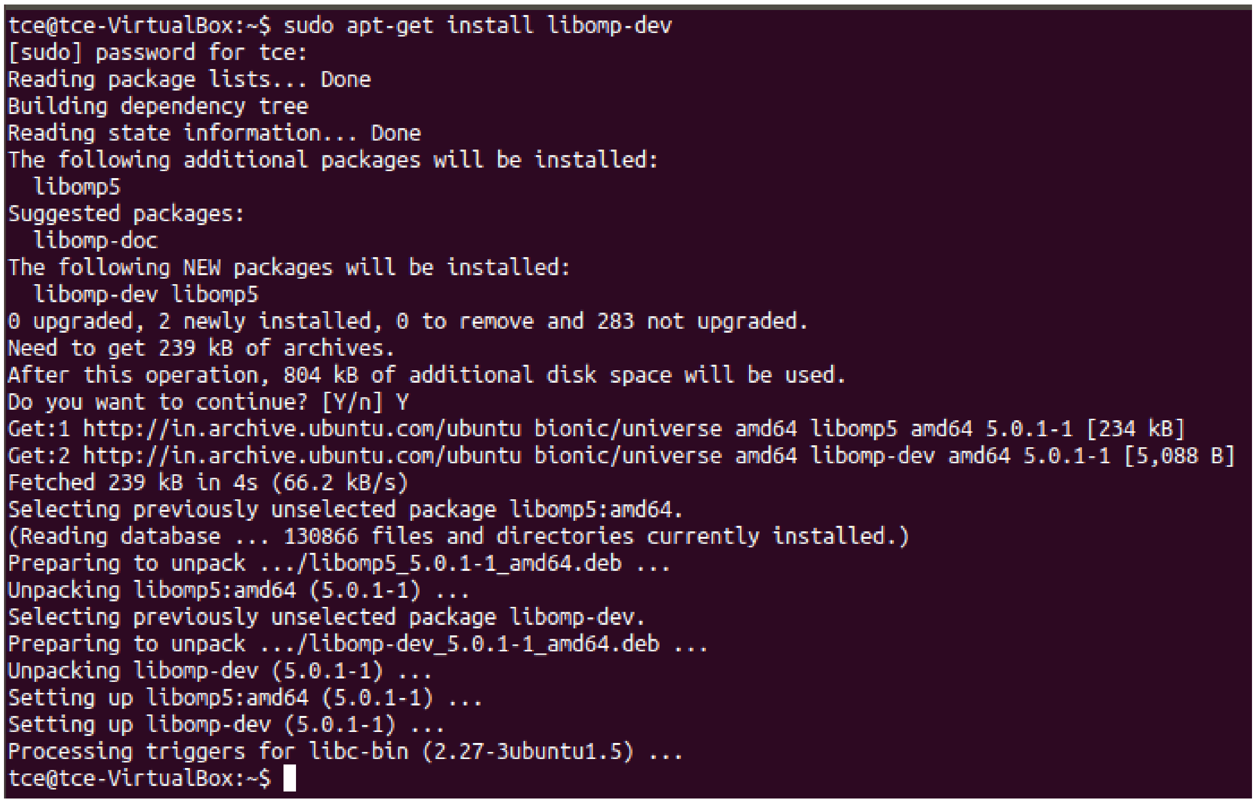
EX:10 **IMPLEMENTATION OF PARALLEL PROGRAMMING USING OPENMP TO PERFORM TASK MANAGEMENT**
23/09/24

Aim:

To implement parallel programming using Openmp.

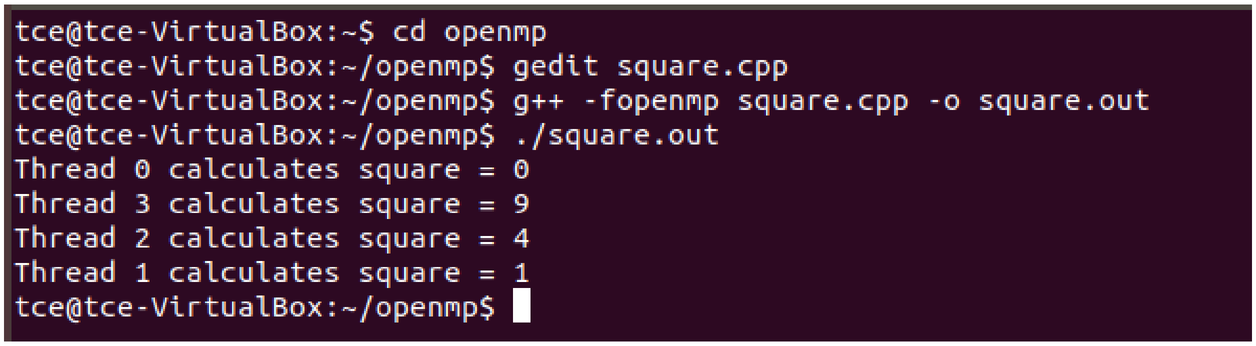
Procedure:

1) Install the necessary libraries in ubuntu virtual machine and run set of programs



```
tce@tce-VirtualBox:~$ sudo apt-get install libomp-dev
[sudo] password for tce:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libomp5
Suggested packages:
  libomp-doc
The following NEW packages will be installed:
  libomp-dev libomp5
0 upgraded, 2 newly installed, 0 to remove and 283 not upgraded.
Need to get 239 kB of archives.
After this operation, 804 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 libomp5 amd64 5.0.1-1 [234 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 libomp-dev amd64 5.0.1-1 [5,088 B]
Fetched 239 kB in 4s (66.2 kB/s)
Selecting previously unselected package libomp5:amd64.
(Reading database ... 130866 files and directories currently installed.)
Preparing to unpack .../libomp5_5.0.1-1_amd64.deb ...
Unpacking libomp5:amd64 (5.0.1-1) ...
Selecting previously unselected package libomp-dev.
Preparing to unpack .../libomp-dev_5.0.1-1_amd64.deb ...
Unpacking libomp-dev (5.0.1-1) ...
Setting up libomp5:amd64 (5.0.1-1) ...
Setting up libomp-dev (5.0.1-1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.5) ...
tce@tce-VirtualBox:~$
```

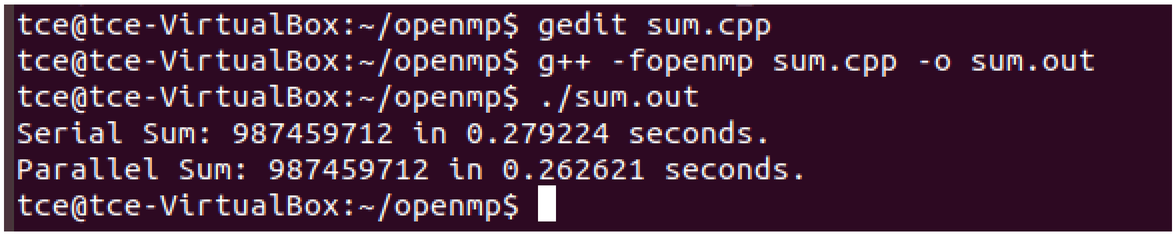
<u>Sample 1: Square</u>



```
tce@tce-VirtualBox:~$ cd openmp
tce@tce-VirtualBox:~/openmp$ gedit square.cpp
tce@tce-VirtualBox:~/openmp$ g++ -fopenmp square.cpp -o square.out
tce@tce-VirtualBox:~/openmp$ ./square.out
Thread 0 calculates square = 0
Thread 3 calculates square = 9
Thread 2 calculates square = 4
Thread 1 calculates square = 1
tce@tce-VirtualBox:~/openmp$
```

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

int main() {
    omp_set_num_threads(4);  // Set the number of threads to 4

    #pragma omp parallel
    {
        int thread_id = omp_get_thread_num();   // Get the thread ID
        int square = thread_id * thread_id;     // Calculate square of the thread ID

        std::cout << "Thread " << thread_id
                  << " calculates square = " << square << std::endl;
    }

    return 0;
}
```

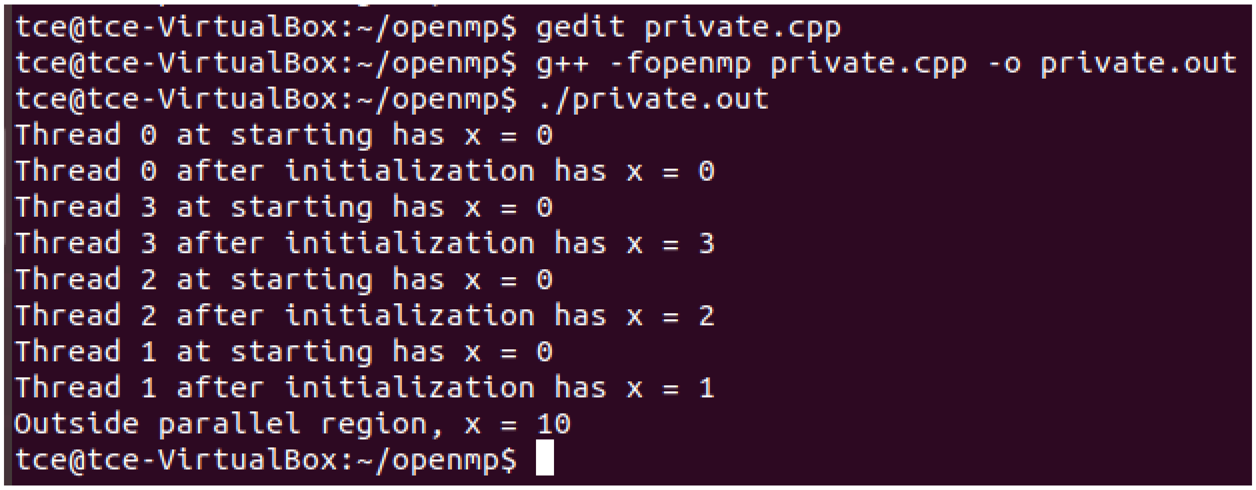
Sample 2: Loop level Parallelism with reduction



```
tce@tce-VirtualBox:~/openmp$ gedit sum.cpp
tce@tce-VirtualBox:~/openmp$ g++ -fopenmp sum.cpp -o sum.out
tce@tce-VirtualBox:~/openmp$ ./sum.out
Serial Sum: 987459712 in 0.279224 seconds.
Parallel Sum: 987459712 in 0.262621 seconds.
tce@tce-VirtualBox:~/openmp$
```

```
#include <omp.h>
#include <iostream>
#include <chrono>

#define THREAD_COUNT 5

using namespace std;

int serial_sum(int n) {
    int sum = 0;
    for (int i = 1; i <= n; i++) {
        sum += i;
    }
    return sum;
}

int parallel_sum(int n) {
    int sum = 0;
    #pragma omp parallel for reduction(+:sum)
    for (int i = 1; i <= n; i++) {
        sum += i;
    }
    return sum;
}

int main() {
    omp_set_num_threads(THREAD_COUNT);

    int n = 100000000;

    auto start_time1 = chrono::high_resolution_clock::now();
    int ser_sum = serial_sum(n);
    auto end_time1 = chrono::high_resolution_clock::now();
    chrono::duration<double> duration1 = end_time1 - start_time1;

    cout << "Serial Sum: " << ser_sum << " in " << duration1.count() << " seconds." << endl;

    auto start_time2 = chrono::high_resolution_clock::now();
    int par_sum = parallel_sum(n);
    auto end_time2 = chrono::high_resolution_clock::now();
    chrono::duration<double> duration2 = end_time2 - start_time2;

    cout << "Parallel Sum: " << par_sum << " in " << duration2.count() << " seconds." << endl;

    return 0;
}
```

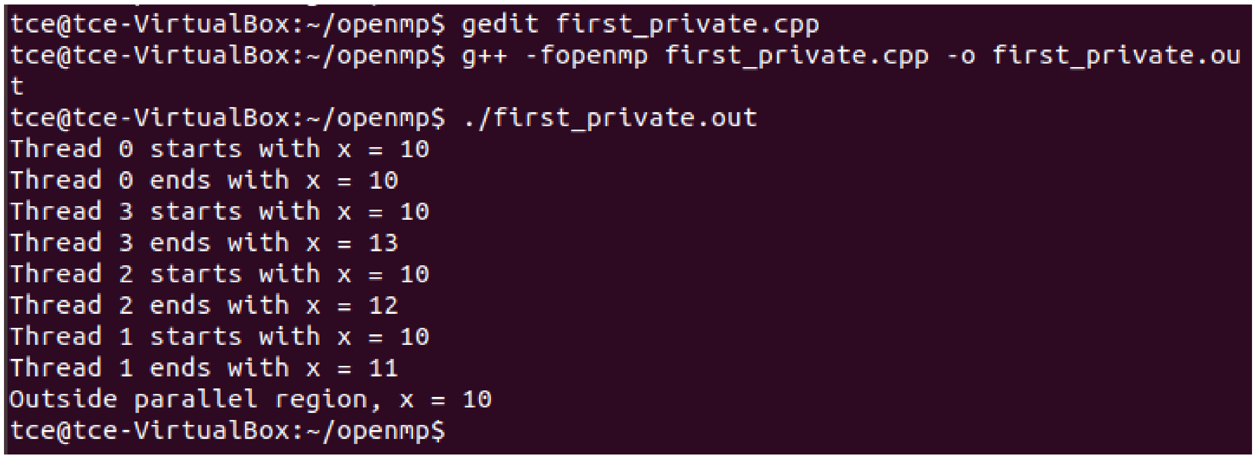
<u>Sample 3 : Using private</u>



```
tce@tce-VirtualBox:~/openmp$ gedit private.cpp
tce@tce-VirtualBox:~/openmp$ g++ -fopenmp private.cpp -o private.out
tce@tce-VirtualBox:~/openmp$ ./private.out
Thread 0 at starting has x = 0
Thread 0 after initialization has x = 0
Thread 3 at starting has x = 0
Thread 3 after initialization has x = 3
Thread 2 at starting has x = 0
Thread 2 after initialization has x = 2
Thread 1 at starting has x = 0
Thread 1 after initialization has x = 1
Outside parallel region, x = 10
tce@tce-VirtualBox:~/openmp$
```

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

using namespace std;

int main() {
    omp_set_num_threads(4);
    int x = 10;

    #pragma omp parallel private(x)
    {
cout << "Thread " << omp_get_thread_num() << " at starting has x = " << x << endl;
        x = omp_get_thread_num();
        cout << "Thread " << omp_get_thread_num() << " after initialization has x = " << x <<
endl;
    }

    cout << "Outside parallel region, x = " << x << endl;

    return 0;
}
```

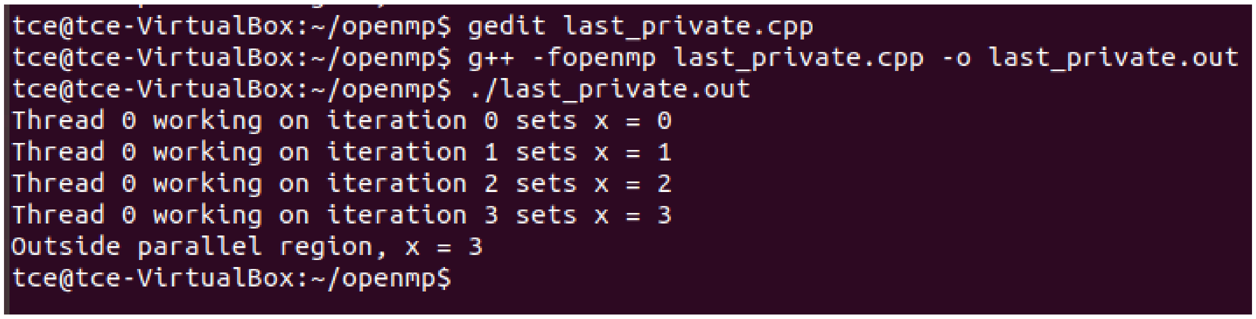
Sample 4: Usage of First private.



```
tce@tce-VirtualBox:~/openmp$ gedit first_private.cpp
tce@tce-VirtualBox:~/openmp$ g++ -fopenmp first_private.cpp -o first_private.ou
t
tce@tce-VirtualBox:~/openmp$ ./first_private.out
Thread 0 starts with x = 10
Thread 0 ends with x = 10
Thread 3 starts with x = 10
Thread 3 ends with x = 13
Thread 2 starts with x = 10
Thread 2 ends with x = 12
Thread 1 starts with x = 10
Thread 1 ends with x = 11
Outside parallel region, x = 10
tce@tce-VirtualBox:~/openmp$
```

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

using namespace std;

int main() {
    omp_set_num_threads(4);
    int x = 10;
    #pragma omp parallel firstprivate(x)
    {
cout << "Thread " << omp_get_thread_num()
             << " starts with x = " << x << endl;
        x += omp_get_thread_num();
        cout << "Thread " << omp_get_thread_num()
             << " ends with x = " << x << endl;
    }

    cout << "Outside parallel region, x = " << x << endl;

    return 0;
}
```

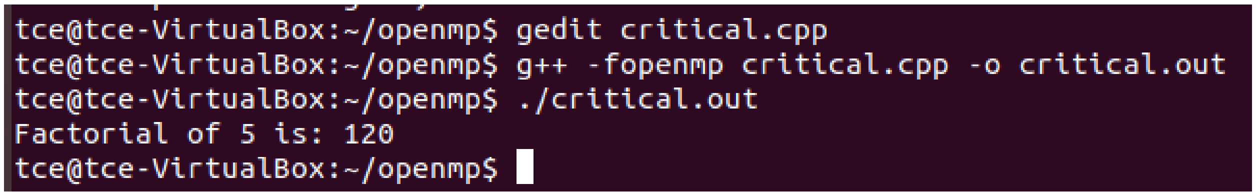
Sample 5: Using last private



```
tce@tce-VirtualBox:~/openmp$ gedit last_private.cpp
tce@tce-VirtualBox:~/openmp$ g++ -fopenmp last_private.cpp -o last_private.out
tce@tce-VirtualBox:~/openmp$ ./last_private.out
Thread 0 working on iteration 0 sets x = 0
Thread 0 working on iteration 1 sets x = 1
Thread 0 working on iteration 2 sets x = 2
Thread 0 working on iteration 3 sets x = 3
Outside parallel region, x = 3
tce@tce-VirtualBox:~/openmp$
```

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

using namespace std;

int main() {
    int x = 10;

    #pragma omp parallel for lastprivate(x)
    for (int i = 0; i < 4; i++) {
        x = i;
        cout << "Thread " << omp_get_thread_num()
             << " working on iteration " << i
             << " sets x = " << x << endl;
    }

    cout << "Outside parallel region, x = " << x << endl;

    return 0;
}
```

Sample 6: Using critical section



```
tce@tce-VirtualBox:~/openmp$ gedit critical.cpp
tce@tce-VirtualBox:~/openmp$ g++ -fopenmp critical.cpp -o critical.out
tce@tce-VirtualBox:~/openmp$ ./critical.out
Factorial of 5 is: 120
tce@tce-VirtualBox:~/openmp$
```

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

using namespace std;

int main() {
    int factorial = 1;
    int n = 5;

    #pragma omp parallel for
    for (int i = 1; i <= n; i++) {
        #pragma omp critical
        {
            factorial *= i;
        }
    }

    cout << "Factorial of " << n << " is: " << factorial << endl;

    return 0;
}
```

Result:

Thus ,the parallel processing have been excuted successfully using Openmp.