CSE 4001

PARALLEL AND DISTRIBUTED COMPUTING

Lab Assessment – 2

L27+L28 | PLBG04
Dr. Narayanan Prasanth

FALL SEMESTER 2021-22

by

SHARADINDU ADHIKARI

19BCE2105

Q1. Develop an OpenMP program to parallelize the matrix vector multiplication.

Code:

```
//Q1. Develop an OpenMP program to parallelize the matrix vector multiplication.

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

//19BCE2105, Sharadindu Adhikari

int a[1000] = {8,4,6,7,1,2,5,3};
int b[1000];

   void Merge(int low, int mid, int high)
   {
     int i, j, k;

     i=low; j=mid+1; k=low;
     while ( i <= mid && j <= high )
       {
         if( a[i] <= a[j] )
           b[k++] = a[i++] ;
         else
           b[k++] = a[j++] ;
       }
     while (i <= mid) b[k++] = a[i++] ;
     while (j <= high) b[k++] = a[j++] ;
     for(k=low; k  <= high; k++) a[k] = b[k];
   }

   void MergeSort(int low, int high)
   {
     int mid;
     if(low < high)
       {
         mid = (low+high)/2 ;
         #pragma omp parallel sections
         {
           #pragma omp section

             {
             MergeSort(low, mid);
```

```
            }
          #pragma omp section
            {
            MergeSort(mid+1, high);
            }
        }
        Merge(low, mid, high);
      }
  }

  void main()
  {
    int n, k;
    clock_t st, et;
    double ts;
    printf("Enter input size: ");
    scanf("%d", &n);
    srand(time(0));
    for(k=0; k < n; k++)
      {
        a[k]=rand()%n;
      }
    st=clock(); MergeSort(0, n-1); et=clock();
    ts=(double)(et-st)/CLOCKS_PER_SEC;

    printf("Sorted Numbers are : \n ");
    for(k=0; k < n; k++) printf("%d ", a[k]);
    printf("\nThe time taken is %e\n", ts);
  }
```

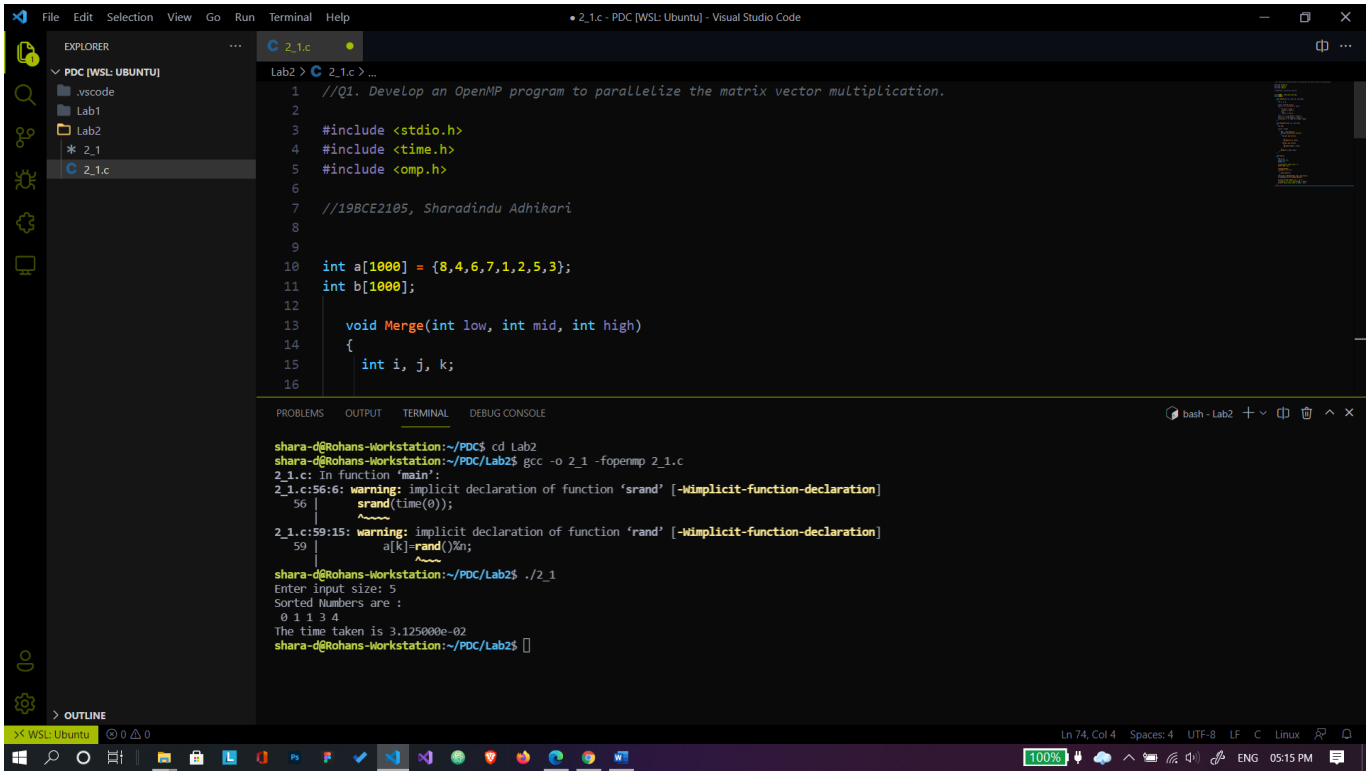
Input:

```
Enter input size: 5
```

Output:

```
shara-d@Rohans-Workstation:~/PDC$ cd Lab2
shara-d@Rohans-Workstation:~/PDC/Lab2$ gcc -o 2_1 -fopenmp 2_1.c
shara-d@Rohans-Workstation:~/PDC/Lab2$ ./2_1
Enter input size: 5
Sorted Numbers are :
 0 1 1 3 4
The time taken is 3.125000e-02
```

Screenshot:



Q2. Parallelize the operations of linear and binary search using OpenMP constructs. Compute the time taken by the above approaches.

Code:

```
//Q2. Parallelize the operations of linear and binary search using OpenMP
constructs. Compute the time taken by the above approaches.

#include <stdio.h>
#include <time.h>
#include <omp.h>
#include<math.h>

//19BCE2105, Sharadindu Adhikari

int linear(int*,int);
void binary(int*,int);
double st,et;

int linear(int* a,int p)
{
```

```
    int f=-1;
    st=omp_get_wtime();
    #pragma omp parallel for
    for(int h=0;h<5;h++)
    {
        if(a[h]==p)
        {
            f++;
        }
    }
    et=omp_get_wtime()-st;
    return f;
}

void binary(int* a,int key)
{
 double end_time,start_time;
 int sep[5],i,j,n=2,left,right,size=5,interval,index,break_value=0,tid;
 left=0;
 right=size-1;

 if(key>=a[left] && key<=a[right])
  {
   while(left!=right)
   {

    start_time = omp_get_wtime();
     printf("left=%d, right=%d, size=%d\n",left,right,size);
     if(size<=n)
      {

       #pragma omp parallel for num_threads(size)
       for(i=0;i<size;i++)
        {
         sep[i]=left+i;
         tid=omp_get_thread_num();
         printf("Thread %d allocated sep[%d]=%d\n",tid,i,sep[i]);
        }
      }

     else
      {
       sep[0]=left;
       interval=ceil((float)size/(float)n);
```

```
      #pragma omp parallel for num_threads(n-1)
      for(i=1;i<=n-1;i++)
       {
        sep[i]=left+interval*i-1;
        tid=omp_get_thread_num();
        printf("Thread %d allocated sep[%d]=%d\n",tid,i,sep[i]);
       }

       sep[n]=right;
      }

    end_time = omp_get_wtime()-start_time;


      for(i=0;i<=n;i++)
       {
         if(key==a[sep[i]])
          {
            index=sep[i];
            printf("Element found at position %d\n",index+1);
            break_value=1;
            break;
          }
         if(key<a[sep[i]])
          {
            right=sep[i];
            if(i!=0)
              left=1+sep[i-1];
            size=right-left+1;
            break;
          }
       }


      if(break_value==1)
        break;
   }
  }

if(left==right || !(key>=a[left] && key<=a[right]))
  printf("Element does not present in the list\n");

printf("Time taken for Binary Search is :%fs\n",end_time);
```

```
}

int main()
{

    int a[5]={11,22,69,99,100};
    int p;
    printf("Enter the key to be searched\n");
    scanf("%d",&p);
    if(linear(a,p)==0)
    printf("Found\n");
    else
    printf("Not Found\n");
    printf("Time taken for Linear Search is: %fs\n",et);
    binary(a,p);

}
```

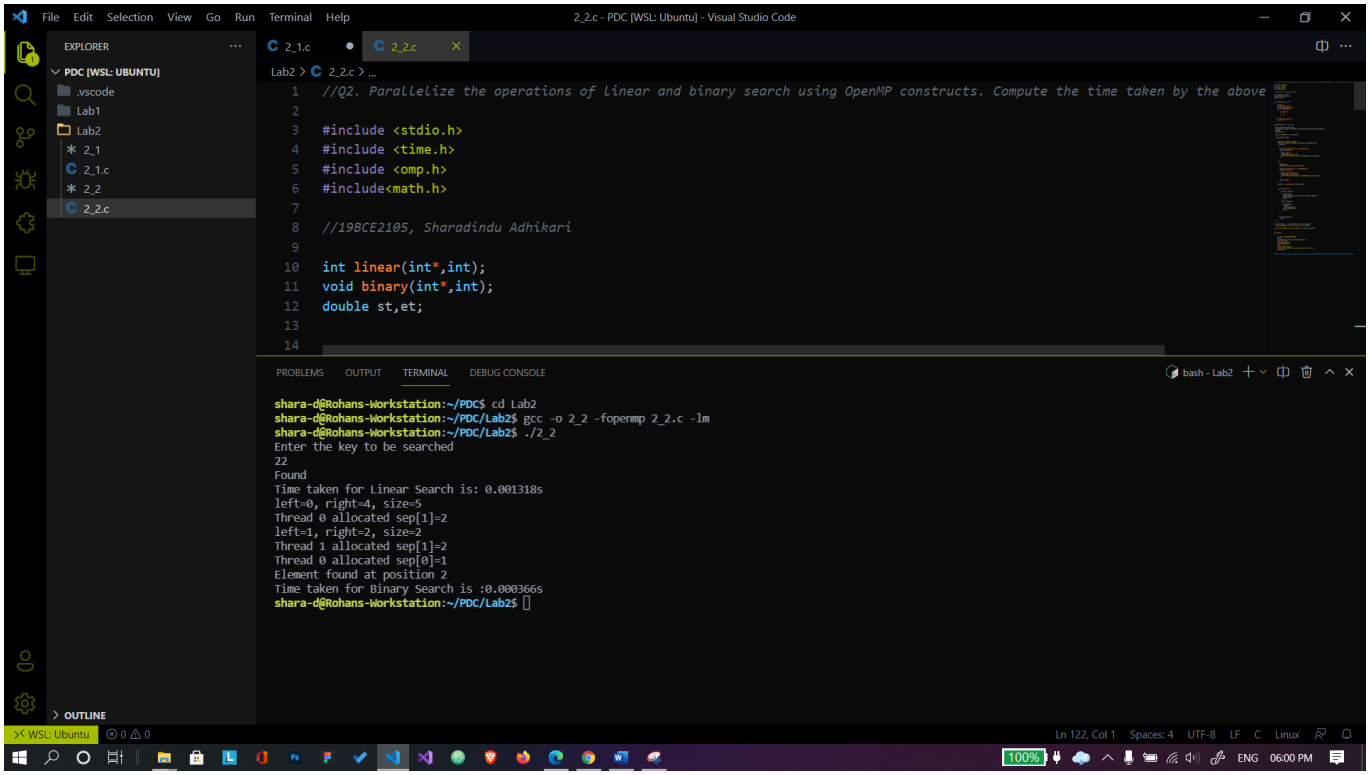
Input:

```
Enter the key to be searched: 22
```

Output:

```
shara-d@Rohans-Workstation:~/PDC/Lab2$ gcc -o 2_2 -fopenmp 2_2.c -lm
shara-d@Rohans-Workstation:~/PDC/Lab2$ ./2_2
Enter the key to be searched
22
Found

Time taken for Linear Search is: 0.001318s
left=0, right=4, size=5
Thread 0 allocated sep[1]=2
left=1, right=2, size=2
Thread 1 allocated sep[1]=2
Thread 0 allocated sep[0]=1
Element found at position 2

Time taken for Binary Search is :0.000366s
```

Screenshot:



Q3. Develop an OpenMP program to perform Matrix multiplications using the appropriate synchronization constructs.

Code and Input:

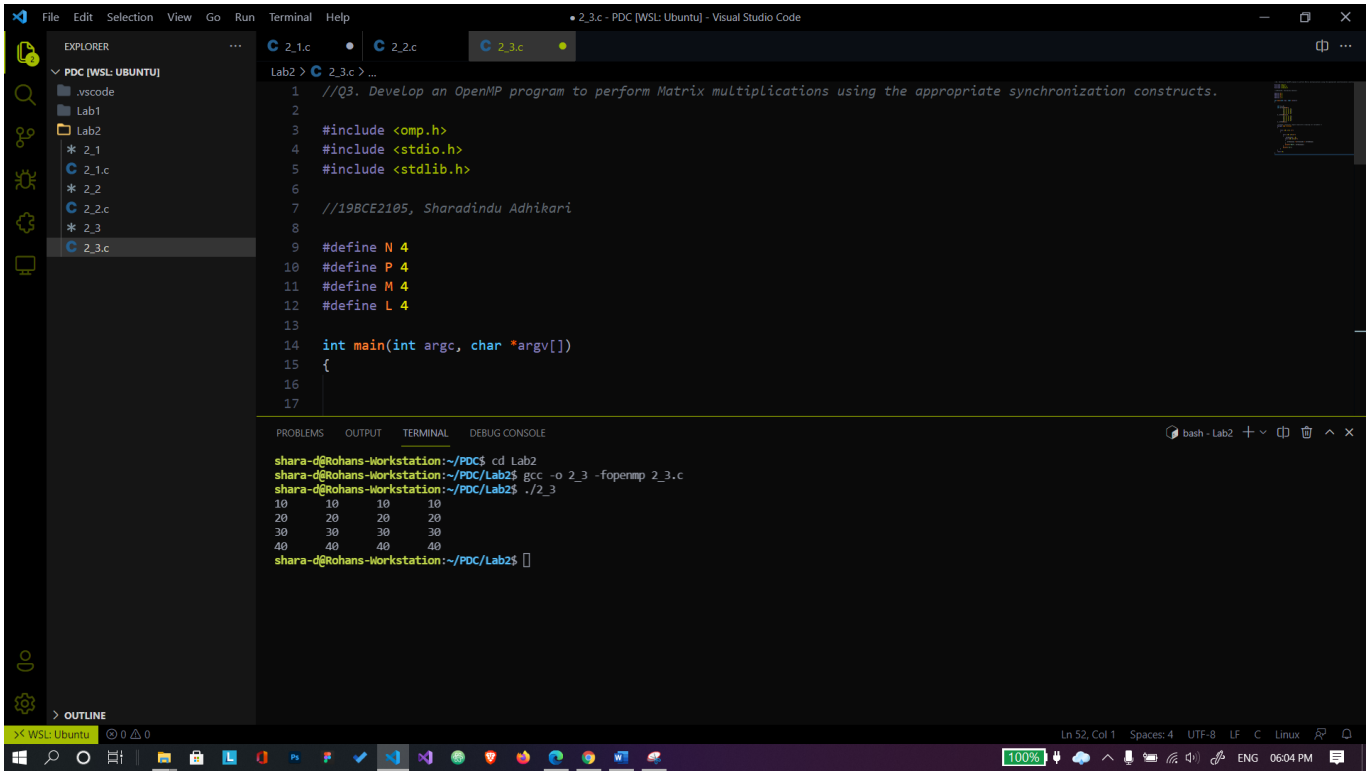
```
//Q3. Develop an OpenMP program to perform Matrix multiplications using the
appropriate synchronization constructs.

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

//19BCE2105, Sharadindu Adhikari

#define N 4
#define P 4
#define M 4
#define L 4

int main(int argc, char *argv[])
{
```

```
    int i,j,k;
    int arr1[N][P]={
            {1, 1, 1, 1},
            {2, 2, 2, 2},
            {3, 3, 3, 3},
            {4, 4, 4, 4}
    }, arr2[M][L]={
            {1, 1, 1, 1},
            {2, 2, 2, 2},
            {3, 3, 3, 3},
            {4, 4, 4, 4}
    }, arr3[N][L];

    /*Create a parallel region explicitly scoping all variables */
    #pragma omp critical
    {

        for( i=0; i<4; i++)
        {

            for( j=0; j<4;j++)
            {
                arr3[i][j] = 0;
               for( k=0; k<4;k++)
               {
                  arr3[i][j] +=arr1[i][k] * arr2[k][j];
               }
               printf("%d\t", arr3[i][j]);
            }
            printf("\n");
        }
    }
    return 0;
}
```

Output:

```
shara-d@Rohans-Workstation:~/PDC/Lab2$ gcc -o 2_3 -fopenmp 2_3.c
shara-d@Rohans-Workstation:~/PDC/Lab2$ ./2_3
10      10      10      10
20      20      20      20
30      30      30      30
40      40      40      40
```

Screenshot:



Q4. Generate a Fibonacci sequence using lock synchronization tool.

Code and Input:

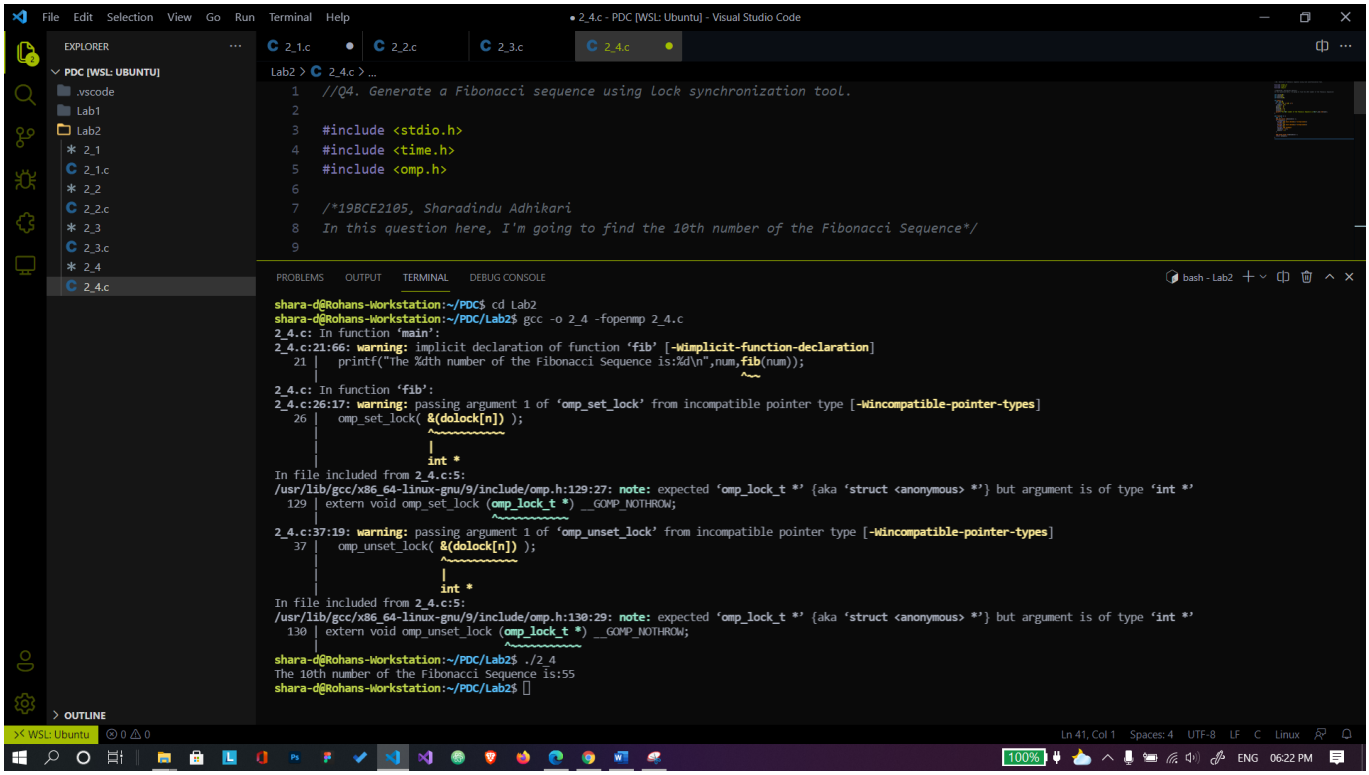
```
//Q4. Generate a Fibonacci sequence using lock synchronization tool.

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

/*19BCE2105, Sharadindu Adhikari
In this question here, I'm going to find the 10th number of the Fibonacci Sequence*/

int value[10];
int done[10];
int dolock[10];

int main() {
  int num=10;
```

```
 for (int i=0; i<=10; i++)
    done[i] = 0;
  done[0] = 0;
  done[1] = 1;
  value[0] = 0;
  value[1] = 1;
  printf("The %dth number of the Fibonacci Sequence is:%d\n",num,fib(num));
}

int fib(int n) {
  int i, j;
  omp_set_lock( &(dolock[n]) );
  if (!done[n]) {
    #pragma omp task shared(i) firstprivate(n)
    i=fib(n-1);
    #pragma omp task shared(j) firstprivate(n)
    j=fib(n-2);
    #pragma omp taskwait
    value[n] = i+j;
    done[n] = 1;
  }

  omp_unset_lock( &(dolock[n]) );
  return value[n];
}
```

Output:

```
shara-d@Rohans-Workstation:~/PDC/Lab2$ gcc -o 2_4 -fopenmp 2_4.c
shara-d@Rohans-Workstation:~/PDC/Lab2$ ./2_4
The 10th number of the Fibonacci Sequence is: 55
```

Screenshot:


File Edit Selection View Go Run Terminal Help
• 2_4.c - PDC [WSL: Ubuntu] - Visual Studio Code
EXPLORER
PDC [WSL: UBUNTU]
.vscode
Lab1
Lab2
2_1
2_1.c
2_2
2_2.c
2_3
2_3.c
2_4
2_4.c
OUTLINE
2_1.c
2_2.c
2_3.c
2_4.c
Lab2 > 2_4.c > ...
1 //Q4. Generate a Fibonacci sequence using lock synchronization tool.
2
3 #include <stdio.h>
4 #include <time.h>
5 #include <omp.h>
6
7 /*19BCE2105, Sharadindu Adhikari
8 In this question here, I'm going to find the 10th number of the Fibonacci Sequence*/
9
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
bash - Lab2
shara-d@Rohans-Workstation:~/PDC$ cd Lab2
shara-d@Rohans-Workstation:~/PDC/Lab2$ gcc -o 2_4 -fopenmp 2_4.c
2_4.c: In function 'main':
2_4.c:21:66: warning: implicit declaration of function 'fib' [-Wimplicit-function-declaration]
21 | printf("The %dth number of the Fibonacci Sequence is:%d\n",num,fib(num));
| ^~~
2_4.c: In function 'fib':
2_4.c:26:17: warning: passing argument 1 of 'omp_set_lock' from incompatible pointer type [-Wincompatible-pointer-types]
26 | omp_set_lock(&(dolock[n]));
| ^~~~~~~~~~~
| |
| int *
In file included from 2_4.c:5:
/usr/lib/gcc/x86_64-linux-gnu/9/include/omp.h:129:27: note: expected 'omp_lock_t *' {aka 'struct <anonymous> *'} but argument is of type 'int *'
129 | extern void omp_set_lock (omp_lock_t *) __GOMP_NOTHROW;
| ^~~~~~~~~~~
2_4.c:37:19: warning: passing argument 1 of 'omp_unset_lock' from incompatible pointer type [-Wincompatible-pointer-types]
37 | omp_unset_lock(&(dolock[n]));
| ^~~~~~~~~~~
| |
| int *
In file included from 2_4.c:5:
/usr/lib/gcc/x86_64-linux-gnu/9/include/omp.h:130:29: note: expected 'omp_lock_t *' {aka 'struct <anonymous> *'} but argument is of type 'int *'
130 | extern void omp_unset_lock (omp_lock_t *) __GOMP_NOTHROW;
| ^~~~~~~~~~~
shara-d@Rohans-Workstation:~/PDC/Lab2$ ./2_4
The 10th number of the Fibonacci Sequence is:55
shara-d@Rohans-Workstation:~/PDC/Lab2$
WSL: Ubuntu
Ln 41, Col 1 Spaces: 4 UTF-8 LF C Linux
100%
ENG 06:22 PM