Aryaman Mishra

19BCE1027

Aim

Consider a text file as input to the program. Write an efficient **OpenMP** program to identify the number of repetitions of words in the file and display it in the terminal. Let the program print the words and its repetitions along with the corresponding threadID. Let the processes pass the number of words it verified to one of the process which prints the same. Use a minimum of 2 processes for the purpose and implement the above process for any four words of your choice.

Code

```
#include <stdio.h>
#include <string.h>
#include <omp.h>
void main()
{
   int count = 0,c = 0, i, j = 0, k, space = 0;
   char str[100], p[50][100], str1[20], ptr1[50][100];
   char *ptr;
   printf("Enter the text file\n");
   scanf(" %[^\n]s", str);
   int nthreads,tid;
   omp_set_num_threads(2);
   #pragma omp parallel private(tid)
   {
          tid=omp_get_thread_num();
   for (i = 0;i<strlen(str);i++)
   {
      if ((str[i] == ' ')||(str[i] == ',' && str[i+1] == ' ')||(str[i] == '.'))
      {
         space++;
      }
   }
```

```
for (i = 0, j = 0, k = 0;j < strlen(str);j++)
{
   if ((str[j] == ' ')||(str[j] == 44)||(str[j] == 46))
   {
      p[i][k] = '\0';
      i++;
      k = 0;
   }
   else
       p[i][k++] = str[j];
}

k = 0;
for (i = 0;i <= space;i++)
{
   for (j = 0;j <= space;j++)
   {
      if (i == j)
      {
         strcpy(ptr1[k], p[i]);
         k++;
         count++;

         break;
      }
      else
      {
         if (strcmp(ptr1[j], p[i]) != 0)
            continue;
         else
            break;
```

```
        }
      }
   }
   for (i = 0;i < count;i++)
   {
      for (j = 0;j <= space;j++)
      {
         if (strcmp(ptr1[i], p[j]) == 0)
            c++;
      }
      printf("%s -> %d times from thread=%d\n", ptr1[i], c,tid);
      c = 0;
      if(tid==0)
      {
           nthreads=omp_get_num_threads();
           printf("Number of threads=%d\n",nthreads);
                    }
   }
}
}
```

Sample Input-Output

Input

learning to code is learning to create and innovate

Output

Learning -> 2 time from thread=(id number)

To -> 2 time from thread=(id number)

Code ->1 time from thread=(id number)

Is -> 1 time from thread=(id number)

Learning -> 1 time from thread=(id number)

Create -> 1 time from thread=(id number)

And -> 1 time from thread=(id number)

Innovate-> 1 time from thread=(id number)

Result Screenshot

```
aryaman@aryaman-VirtualBox:~/Desktop/19BCE1027PDC$ gcc -o fat -fopenmp fat.c
aryaman@aryaman-VirtualBox:~/Desktop/19BCE1027PDC$ ./fat
Enter the text file
aryaman mishra is the best aryaman i know
aryaman -> 2 times from thread=1
mishra -> 1 times from thread=1
is -> 1 times from thread=1
the -> 1 times from thread=1
best -> 1 times from thread=1
i -> 1 times from thread=1
know -> 1 times from thread=1
aryaman -> 2 times from thread=0
Number of threads=2
mishra -> 1 times from thread=0
Number of threads=2
is -> 1 times from thread=0
Number of threads=2
the -> 1 times from thread=0
Number of threads=2
best -> 1 times from thread=0
Number of threads=2
i -> 1 times from thread=0
Number of threads=2
know -> 1 times from thread=0
Number of threads=2
 -> 7 times from thread=0
Number of threads=2
 k -> 0 times from thread=0
Number of threads=2
 -> 7 times from thread=0
Number of threads=2
 -> 0 times from thread=0
Number of threads=2
 -> 7 times from thread=0
Number of threads=2
aryaman@aryaman-VirtualBox:~/Desktop/19BCE1027PDC$
```

Output Verified.