Kennesaw State University

Parallel and Distributed Computing

Project - Pthread

Given two character strings s1 and s2. Write a Pthread program to find out the number of substrings, in string s1, that is exactly the same as s2.

For example, suppose number_substring(s1, s2) implements the function, then
number_substring(“abcdab”, “ab”) = 2,
number_substring(“aaa”, “a”) = 3,
number_substring(“abac”, “bc”) = 0.

The size of s1 and s2 (n1 and n2) as well as their data are input by users. Assume that
n1 mod NUM_THREADS = 0
and
n2 < n1/NUM_THREADS.

The following is a sequential solution of the problem. read_f() reads the two strings from a file named “string.txt and num_substring() calculates the number of substrings.

https://github.com/kevinsuo/CS4504/blob/main/project-pthread.c

----

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define MAX 10240

int total = 0;
int n1,n2;
char *s1,*s2;
FILE *fp;

int readf(FILE *fp)
{
      if((fp=fopen("strings.txt", "r"))==NULL){
                printf("ERROR: can't open string.txt!\n");
                return 0;
      }
      s1=(char *)malloc(sizeof(char)*MAX);
      if(s1==NULL){
                printf("ERROR: Out of memory!\n");
                return -1;
      }
      s2=(char *)malloc(sizeof(char)*MAX);
      if(s1==NULL){
                printf("ERROR: Out of memory\n");
                return -1;
```

```
        }
        /*read s1 s2 from the file*/
        s1=fgets(s1, MAX, fp);
        s2=fgets(s2, MAX, fp);
        n1=strlen(s1);  /*length of s1*/
        n2=strlen(s2)-1; /*length of s2*/
        if(s1==NULL || s2==NULL || n1<n2)  /*when error exit*/
                return -1;
}

int num_substring(void)
{
        int i,j,k;
        int count;

        for (i = 0; i <= (n1-n2); i++){
                count=0;
                for(j = i,k = 0; k < n2; j++,k++){  /*search for the next string of size of n2*/
                        if (*(s1+j)!=*(s2+k)){
                                break;
                        }
                        else
                                count++;
                        if(count==n2)
                                total++;            /*find a substring in this step*/
                }
        }
        return total;
}

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

        readf(fp);
        count = num_substring();
        printf("The number of substrings is: %d\n", count);
        return 1;
}
```

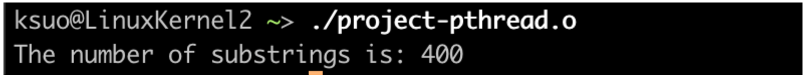
You can find an example of the “string.txt” here:

string.txt: https://github.com/kevinsuo/CS3502/blob/master/strings.txt

To compile the program with Pthread, use:

```
$ gcc project-pthread.c -o project-pthread.o -pthread
```

Current output:



```
ksuo@LinuxKernel2 ~> ./project-pthread.o
The number of substrings is: 400
```

(Different text files output is also different. For the strings.txt, the output is 55)

Download the string.txt:
$ wget https://raw.githubusercontent.com/kevinsuo/CS3502/master/strings.txt

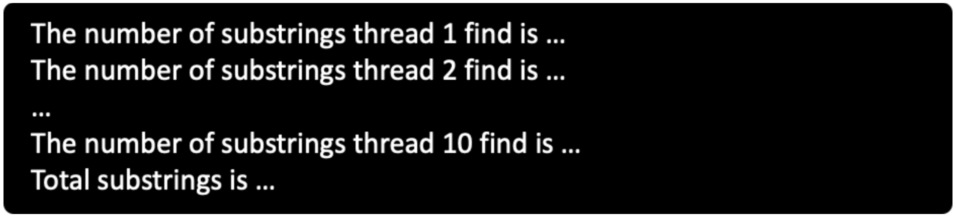
Write a parallel program using Pthread based on this sequential solution. Please set the thread number as 10 in your code. You can start with this template code:
https://github.com/kevinsuo/CS4504/blob/main/parallel-template.c

To compile the program with Pthread, use:

```
$ gcc file.c -o file.o -pthread
```

Here file refers to your source code name.

Expected output:



```
The number of substrings thread 1 find is …
The number of substrings thread 2 find is …
…
The number of substrings thread 10 find is …
Total substrings is …
```

HINT: Strings s1 and s2 are stored in a file named “string.txt”. String s1 is evenly partitioned for *NUM_THREADS* threads to concurrently search for matching with string s2. After a thread finishes its work and obtains the number of local matchings, this local number is added into a global variable showing the total number of matched substrings in string s1. Finally, this total number is printed out. Please make sure the number of substrings of parallel program is the same as the serial program.

Submission

Submit your assignment through D2L using the appropriate link. Please submit the ***source code***, ***output screenshot of your parallel code*** and ***a report describe your code logic***.