Kennesaw State University

Parallel and Distributed Computing

Project - Pthread

Instructor: Kun Suo
Points Possible: 100
Difficulty: ★★★★☆

S1= 🍈🍉🦉🦉❤️🦉🥭🍒🥭🫐❤️🦉

S2= ❤️🦉

Emoji is popular in today's world. Given two emoji strings s1 and s2. Write a Pthread program to find out the number of subEmojiStrings, in string s1, that is exactly the same as s2.

For example, suppose number_subEmojiStrings(s1, s2) implements the function, then
number_subEmojiStrings("🥎🍈🍉🍊🎊🍈🍉🥭🍈", "🍈🍉") = 2,
number_subEmojiStrings("🥎🥎🥎", "🥎") = 3,
number_subEmojiStrings("💚❤️💙❤️", "💚💙") = 0.

The size of s1 and s2 (n1 and n2) as well as their data are input by user file. 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 emoji strings from a file named "emoji.txt and num_subEmojiString() calculates the number of substrings. Here s1 is a string with thousands of emojis and s2 = "❤️🦉".

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

---

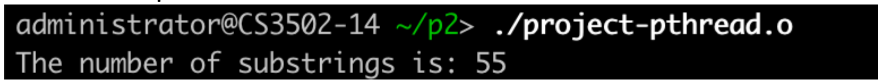
```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define MAX 102400

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

```
FILE *fp;

int readf(FILE *fp)
{
	if((fp=fopen("emoji.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); /*length of s2*/
	if(s1==NULL || s2==NULL || n1<n2)  /*when error exit*/
		return -1;
}

int num_subEmojiString(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_subEmojiString();
	printf("The number of substrings is: %d\n", count);
	return 1;
}
```

You can find an example of the “emoji.txt” here:
https://raw.githubusercontent.com/kevinsuo/CS4504/main/emoji.txt

To compile the program with Pthread, use:
```
$ gcc project-pthread.c -o project-pthread.o -pthread
```

Current output:



```
administrator@CS3502-14 ~/p2> ./project-pthread.o
The number of substrings is: 55
```

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

Download the emoji.txt:

$ wget https://raw.githubusercontent.com/kevinsuo/CS4504/main/emoji.txt

Write a parallel program using Pthread based on this sequential solution. Please set the thread number as **20** in your code. You can start with this template code:
https://github.com/kevinsuo/CS4504/blob/main/parallel-template.c

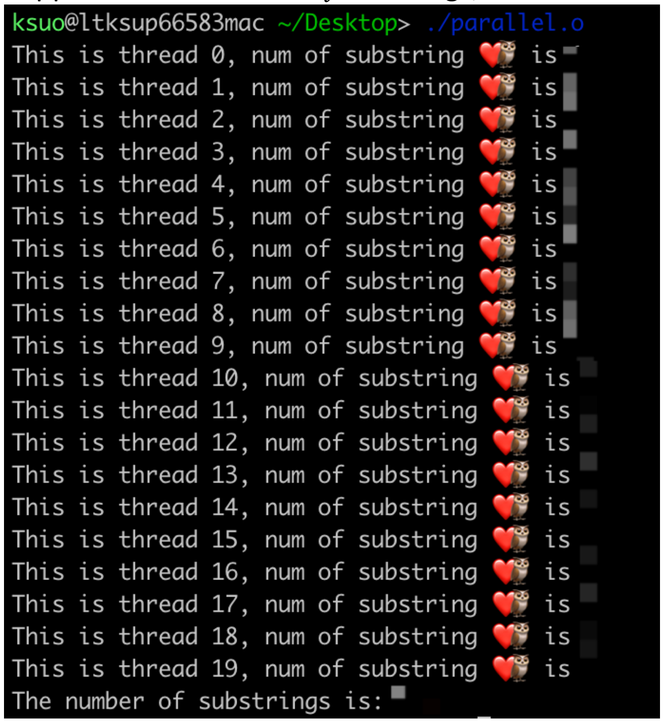
To compile the program with Pthread, use:

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

Here file refers to your source code name.

Expected output (the thread order can be random):

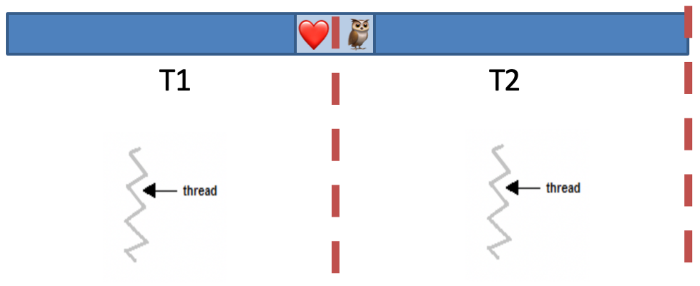
Suppose thread i finds n_i substrings, the total number of substrings should be equal to $\sum_{i=0}^{19} n_i$



```
ksuo@ltksup66583mac ~/Desktop> ./parallel.o
This is thread 0, num of substring ❤️🦉 is [redacted]
This is thread 1, num of substring ❤️🦉 is [redacted]
This is thread 2, num of substring ❤️🦉 is [redacted]
This is thread 3, num of substring ❤️🦉 is [redacted]
This is thread 4, num of substring ❤️🦉 is [redacted]
This is thread 5, num of substring ❤️🦉 is [redacted]
This is thread 6, num of substring ❤️🦉 is [redacted]
This is thread 7, num of substring ❤️🦉 is [redacted]
This is thread 8, num of substring ❤️🦉 is [redacted]
This is thread 9, num of substring ❤️🦉 is [redacted]
This is thread 10, num of substring ❤️🦉 is [redacted]
This is thread 11, num of substring ❤️🦉 is [redacted]
This is thread 12, num of substring ❤️🦉 is [redacted]
This is thread 13, num of substring ❤️🦉 is [redacted]
This is thread 14, num of substring ❤️🦉 is [redacted]
This is thread 15, num of substring ❤️🦉 is [redacted]
This is thread 16, num of substring ❤️🦉 is [redacted]
This is thread 17, num of substring ❤️🦉 is [redacted]
This is thread 18, num of substring ❤️🦉 is [redacted]
This is thread 19, num of substring ❤️🦉 is [redacted]
The number of substrings is: [redacted]
```

HINT: Strings s1 and s2 are stored in a file named “emoji.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.

HINT: A corner case. When search “❤️🦉” using multiple threads, please do not ignore corner case like below. The substring “❤️🦉” could be split into two parts when searched by two threads. Make sure your program can deal with it.



Submission

Submit your assignment file through D2L using the appropriate link.
The submission must include the ***source code***, and ***a report describe your code logic***. ***Output screenshot of your code*** should be included in the report.