CS 3502
Operating Systems

Project 2 Lab

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Parallel and Distributed Computing

Project 2

- Read and write parallel program using Pthread

- Learn to use Pthread functions

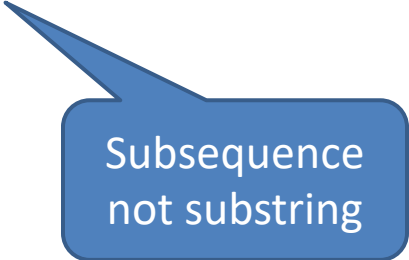- User level projects, not kernel code

Project 2

- 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.

- https://github.com/kevinsuo/CS3502/blob/master/project-pthread.c

Project 2

- number_substring("abcdab", "ab") = 2,

- number_substring("aaa", "a") = 3,

- number_substring("abac", "bc") = 0.

Subsequence not substring

Project 2

- Input file:

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

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

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

```c
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(s2==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;
    return 0;
}
```

```c
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;
}
```

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

    readf(fp);
    count = num_substring();
    printf("The number of substrings is: %d\n", count);
    return 1;
}
```
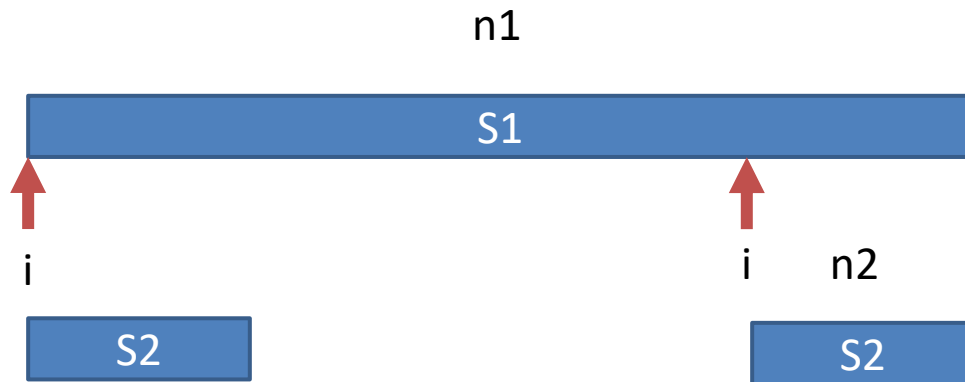
n1

S1

i   i   n2

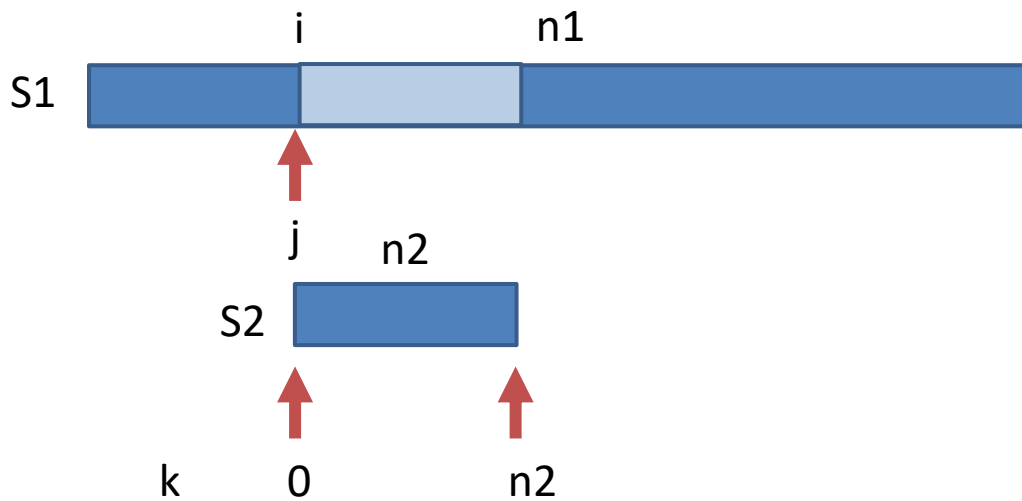S2   S2

```
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;
}
```


S1

i   n1

j   n2

S2

k   0   n2
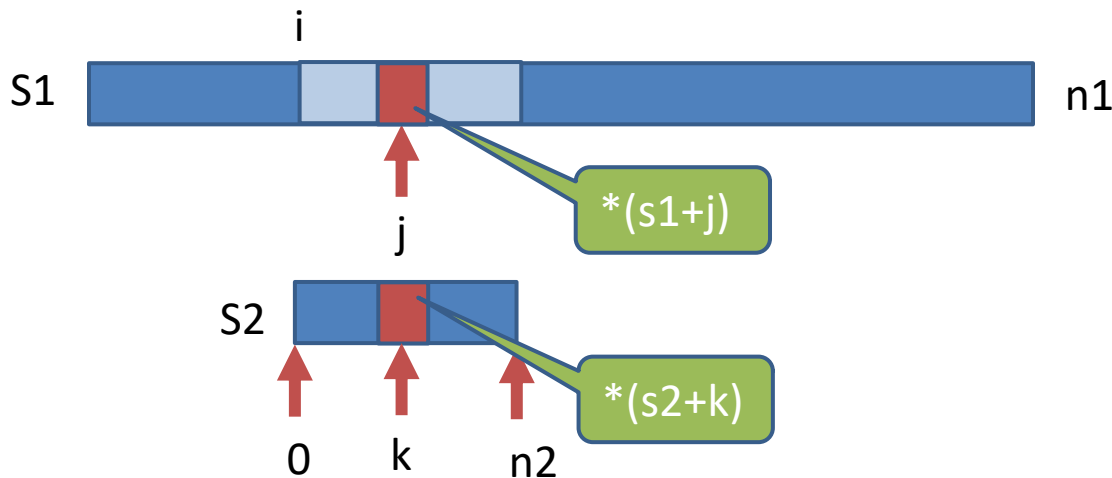
```
int num_substring(void)
{
    int i,j,k;
    int count;

    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;
}
```


S1 ... i ... n1

*(s1+j)

j

S2

*(s2+k)

0   k   n2

```c
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;
}
```

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

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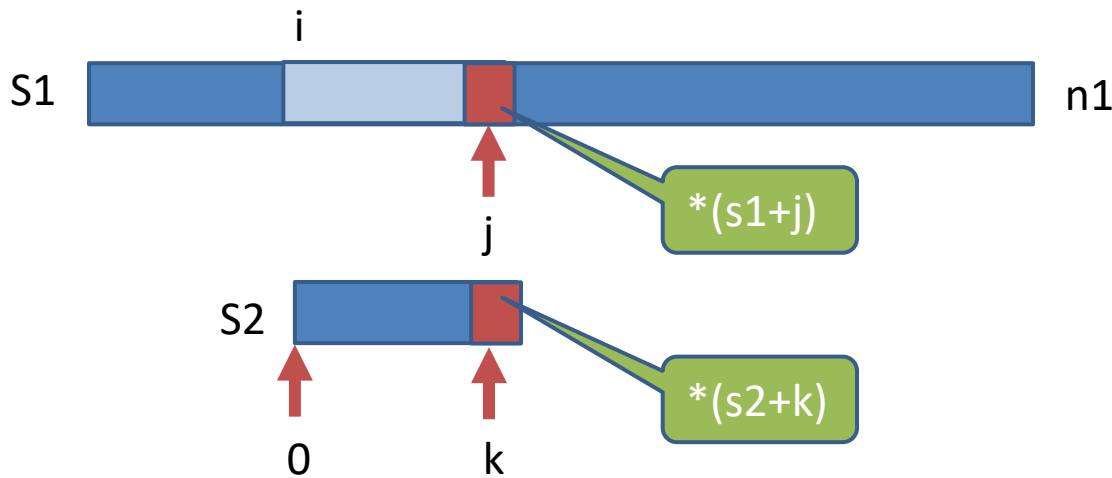
i

S1   n1

*(s1+j)

j

S2

*(s2+k)

0   k

Project 2

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

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

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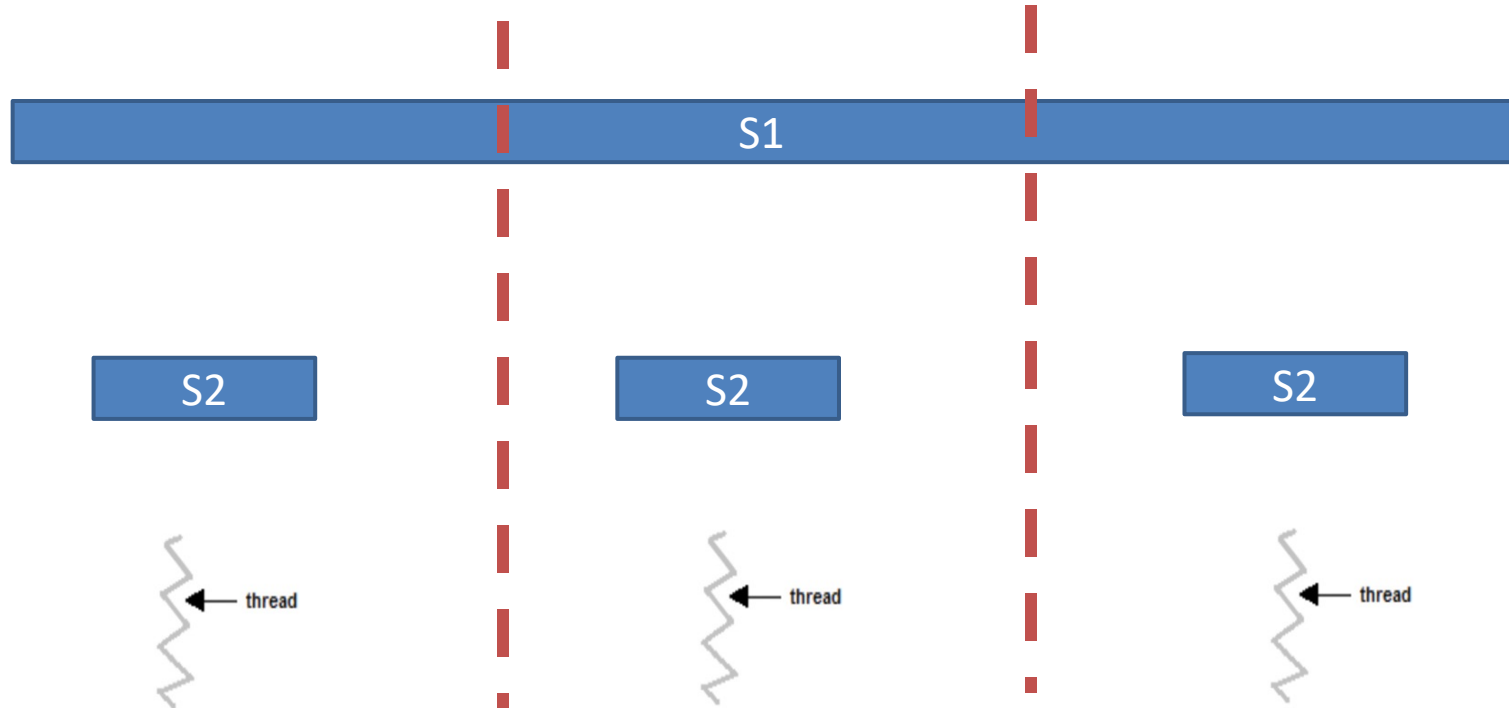
(Different text files output is also different. For the emoji.txt, the output is 55)

```
ksuo@LinuxKernel2 ~> ./project-pthread.o
The number of substrings is: 320
```

Project 2

- Write a parallel program using Pthread based on this sequential solution.

Project 2

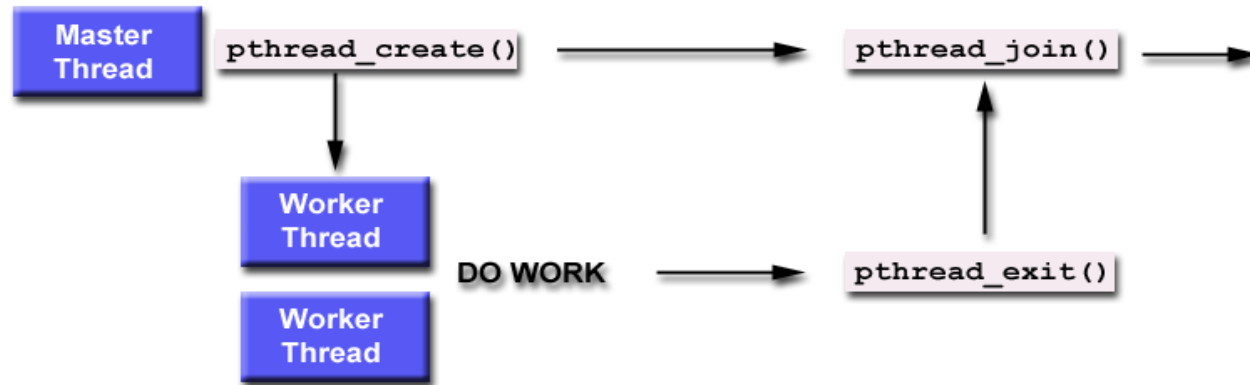**pthread_create** **(thread, attr, start_routine, arg)**

- **creates a thread and makes it executable;**
- **1st parameter: pointer to the thread**
- **2rd parameter: set attributes to threads, usually NULL**
- **3rd parameter: the function for the thread to run**
- **4th parameter: parameter for thread function**

pthread_exit **(status)**

• **If** main() **finishes before the threads it has created, and exists with the** pthread_exit(), **the other threads will continue to execute. Otherwise, they will be automatically terminated when** main() **finishes**

Project 2

```
Master        pthread_create()                    pthread_join()
Thread

              Worker
              Thread
                          DO WORK           pthread_exit()
              Worker
              Thread
```

Main()   thread()
{   {

    //create multiple threads   //do something
    pthread_create   }

    //wait thread to finish in main
     pthread_join

    //sum up the number from each thread
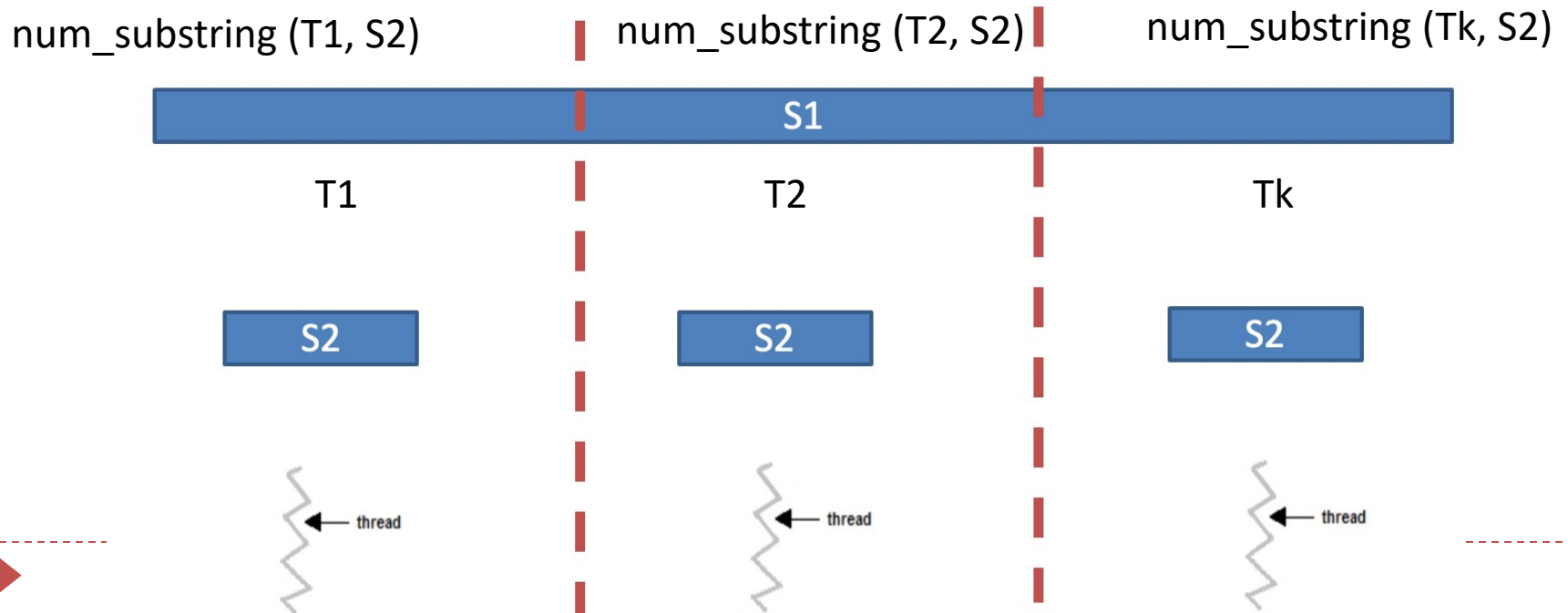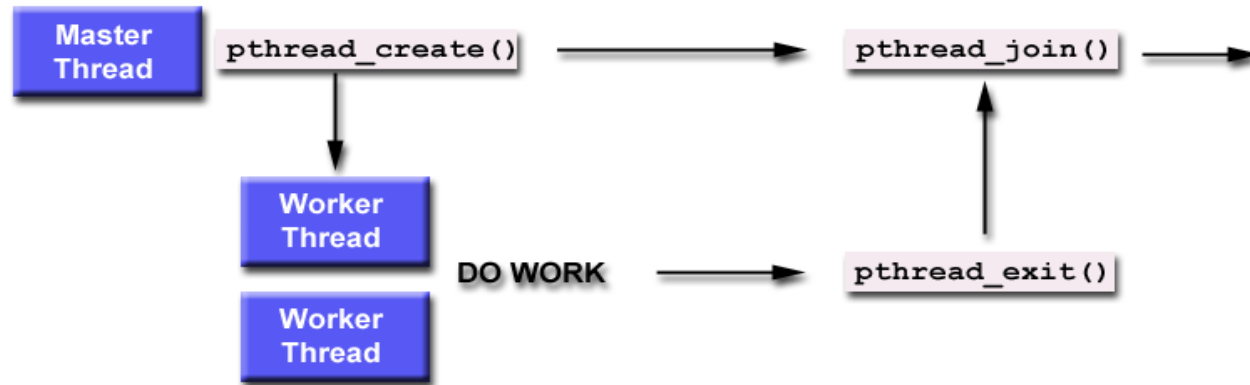   sum = num[0]+ num[1]+ … num[k]
}

Project 2

num_substring (T1, S2)   num_substring (T2, S2)   num_substring (Tk, S2)

Corner Case in Project 2

num_substring (T1, S2)   num_substring (T2, S2)   num_substring (Tk, S2)

| | i | s | S1 | |

T1   T2   Tk

S2   S2   S2

thread   thread   thread
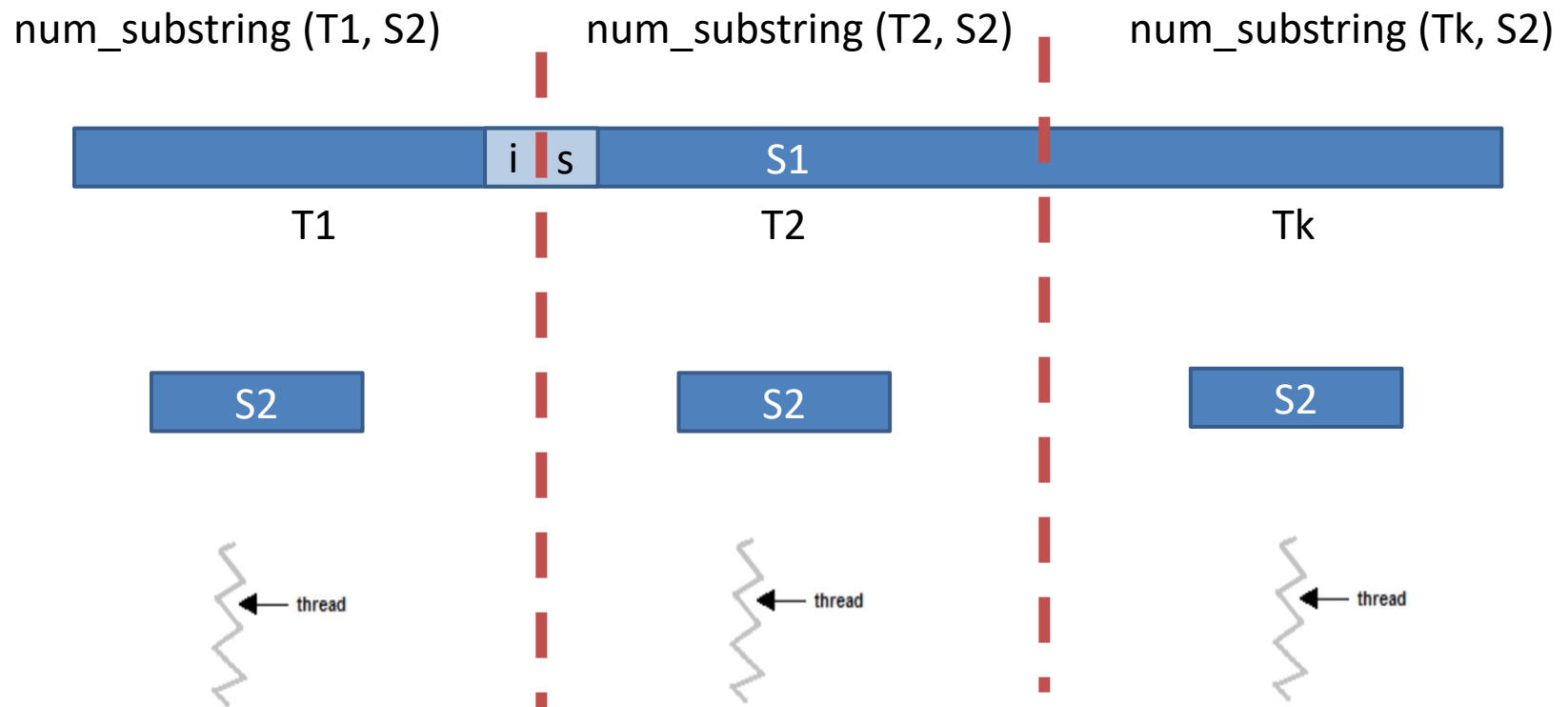
Verify whether your parallel thread is correct

- Modify the strings.txt by yourself

- Compare the sequential and parallel program results that whether they are the same

```
ksuo@LinuxKernel2 ~> ./project-pthread.o
The number of substrings is: 320
ksuo@LinuxKernel2 ~> ./project-pthread-parallel.o
This is thread 0
This is thread 2
This is thread 3
This is thread 1
This is thread 4
The number of substrings is: 320
```

Submission

1. source code

2. output screenshot of your parallel code

3. a report describe your code logic