Artem Tolstov

CSC 415

4/7/2015

Homework 4

The 2 programs I function almost identically with the exception that the win32 version uses createThread() and the posix version replaces that functionality with the pthread_create(). The posix version also requires that it be compiled with –pthread command.

Both programs take an argument from the command line. A check is ran to make sure there are 2 arguments comings in(the name of the program and the name of the text file it will be analyzing. The program will correct the user by displaying correct usage if the proper syntax is not followed.

Provided the correct syntax is followed the programs create the global variables of the buffer and the characterCount array in order to store the information. The length of the string in the buffer is determined and that number is divided by the number of threads that particular version of the program will be running with. That information is put into the struct called threadParams. These structs are passed to the thread, the functionality of which differs depending on whether it is a posix program or win32.

The threads themselves are called asciiCounter. AsciiCounter runs a for loop which increments each of the values in the charCounts array if that letter has appeared in the buffer, charCounts is two-dimensional, the first dimension equaling the particular thread. A final if statement in asciiCounter executes if it is in the final thread. This if statement is necessary if the length of the string in buffer is not evenly divisible by the number of threads and as a result a number of characters at the end of the buffer may not have been recorded. This if statement compensates by adding those remaining characters to the last dimension of charCounts array.

Finally the threads return to the main function in which a final function called printOccurences executes. PrintOccurences adds the values of charCounts various dimensions into one dimension of that array and then executes a for loop which prints each the number of occurrences of each character, printing the character if it is printable and the hex value if it is unprintable, as per the homework instructions.

These are the run times of the 4 different versions of the program in win32 from powershell (edited for readability):

```
PS C:\Users\czar__000\Desktop\csc415\homework4> Measure-Command{.\win32-
8thread test.txt}

TotalMilliseconds : 14.4247

PS C:\Users\czar__000\Desktop\csc415\homework4> Measure-Command{.\win32-
4thread test.txt}

TotalMilliseconds : 13.2481

PS C:\Users\czar__000\Desktop\csc415\homework4> Measure-Command{.\win32-
2thread test.txt}

TotalMilliseconds : 13.7847

PS C:\Users\czar__000\Desktop\csc415\homework4> Measure-Command{.\win32-
1thread test.txt}

TotalMilliseconds : 14.7307
```

It appears that the 2 threaded and 4 threaded versions of this program execute the fastest, this computer has 4 cores. The numbers are too close to map but it might be a parabolic rate of acceleration.

These are the values for the posix version of the program acquired using the time command (edited for readability):

```
real    0m0.165s           real    0m0.099s

user    0m0.004s           user    0m0.000s

sys     0m0.056s           sys     0m0.032s

for 1 thread               for 2 thread


real  0m0.110s             real     0m0.117s

user  0m0.008s             user     0m0.000s

sys   0m0.028s             sys      0m0.040s

for 4 thread               for 8 thread
```

Again the 2 thread and 4 thread processes are faster following the same pattern, however the times are much different. This is running on a significantly slower virtual machine that only utilizes 1 core.

The following is the win32 4 thread version of the C code:

```
//Author: Artem Tolstov

//Purpose: CSC415 HW4

//Function: Takes name of file from command line and counts the number of
ascii characters using

//                predefined number of threads then prints to the screen
occurences of said characters.




#include <stdio.h>

#include <stdlib.h>

#include <windows.h> //for threads

#define BUFFER_SIZE 65536

#define NUM_THREADS 1

void printOccurences(int array[NUM_THREADS][128]);

int charCounts[NUM_THREADS][128];

char buffer[BUFFER_SIZE];

struct threadParams

{

      int threadIndex;

      int stringLengthInBuffer;


};

DWORD WINAPI asciiCounter (LPVOID param);


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

{

    //Declarations

    FILE *filePointer;

    int i,j,x;

      int partition;
      struct threadParams params[NUM_THREADS];
      DWORD ThreadId;
      HANDLE ThreadHandle;



    //Makes sure there are 2 arguments in the command line.
    if (argc != 2 )
      {
            printf( "proper usage: %s filename.extension", argv[0] );
      }
    else
      {
            //Reports if file does not exist.
            filePointer = fopen( argv[1], "r" );

            if (filePointer == 0)
            {
                  printf ("Could not open file\n");
            }
            else
            {
                i = 0;
                  while (((x = fgetc(filePointer)) != EOF) && (i <
BUFFER_SIZE))
                  {
                        //Put next character into i position of buffer,
increment i.
                      buffer[i] = x; i++;
                        //printf( "%c", x);
                  }
                  fclose( filePointer );
                  //For debugging

                  //printf( "\nbuffer contains: %s\n", buffer );

                  printf("length of string in file is: %d\n",
strlen(buffer));



                  //Initialize array charCounts to 0

                  for(i = 0; i < NUM_THREADS; i++)

                  {

                        for(j = 0; j < 128; j++)

                        {

                              charCounts[i][j] = 0;

                        }

                  }


                  //Sends params with thread indexi to threads

                  for(i = 0; i < NUM_THREADS; i++)

                  {

                        params[i].threadIndex = i;

                        params[i].stringLengthInBuffer = strlen(buffer);

                        ThreadHandle = CreateThread(NULL, 0, asciiCounter,
&params[i], 0, &ThreadId);


                  }


                  //waits for threads to finish

                  if (ThreadHandle != NULL)

                  {

                        WaitForSingleObject(ThreadHandle, INFINITE);

                  }
                  CloseHandle(ThreadHandle);

                  printOccurences(charCounts);

            }

      }





}


//Prints letter if printable, Hex value if unprintable character.

//Combines the values of the different threads into [0][] section of the
array.

void printOccurences(int array[NUM_THREADS][128])

{

    int i,j, holdIt;

    for(i = 0; i < 128; i++)

    {

        //holdIt = array[0][i];

            //array[0][i] = 0;

        for(j = 1; j < NUM_THREADS; j++)

        {

           array[0][i] +=  array[j][i];

        }

        //array[0][i] = array[0][i] + holdIt;

    }


    for(i = 0; i < 128; i++)

    {

        if((i > 32) && (i != 127))

        {
            printf("%d occurrences of %c\n", array[0][i], (char)i);
        }
        else
        {
            printf("%d occurrences of 0x%x\n", array[0][i], i);
        }
    }
}


DWORD WINAPI asciiCounter (LPVOID param)
{
      struct threadParams thisThreadsParams = *(struct threadParams*)param;
      int i;
      //printf("asciiCounter has threadIndex = %d\n",
thisThreadsParams.threadIndex);
      for(i = thisThreadsParams.threadIndex *
(thisThreadsParams.stringLengthInBuffer/NUM_THREADS); i <
(thisThreadsParams.threadIndex+1)*(thisThreadsParams.stringLengthInBuffer/NUM
_THREADS); i++)
      {
            //printf("%c is value of %d\n", buffer[i], (int)buffer[i]);
            charCounts[thisThreadsParams.threadIndex][(int)buffer[i]]++;
            //Keeps the null characters in the buffer from writing
      }
      //final check to count remaining characters due to truncation in
division
      if(thisThreadsParams.threadIndex + 1 == NUM_THREADS)
      {

      if(NUM_THREADS*(thisThreadsParams.stringLengthInBuffer/NUM_THREADS) !=
thisThreadsParams.stringLengthInBuffer)
            {
                  for(i =
NUM_THREADS*(thisThreadsParams.stringLengthInBuffer/NUM_THREADS); i <
thisThreadsParams.stringLengthInBuffer; i++)
```

```
                {

        charCounts[thisThreadsParams.threadIndex][(int)buffer[i]]++;
                }
            }
        }

        return 0;
    }
```

The following is the execution the 4 threaded version of the win32 version of the code using test.txt as the sample file. The file contains the text “Hello World!!!!!!!!!!!!!” over and over so the file is about 64kb in size.

```
C:\Users\czar__000\Desktop\csc415\homework4>win32-4thread test.txt

length of string in file is: 65536

0 occurrences of 0x0

0 occurrences of 0x1

0 occurrences of 0x2

0 occurrences of 0x3

0 occurrences of 0x4

0 occurrences of 0x5

0 occurrences of 0x6

0 occurrences of 0x7

0 occurrences of 0x8

0 occurrences of 0x9

0 occurrences of 0xa

0 occurrences of 0xb

0 occurrences of 0xc

0 occurrences of 0xd

0 occurrences of 0xe

0 occurrences of 0xf

0 occurrences of 0x10

0 occurrences of 0x11

0 occurrences of 0x12

0 occurrences of 0x13

0 occurrences of 0x14

0 occurrences of 0x15

0 occurrences of 0x16

0 occurrences of 0x17

0 occurrences of 0x18

0 occurrences of 0x19
```

```
0 occurrences of 0x1a

0 occurrences of 0x1b

0 occurrences of 0x1c

0 occurrences of 0x1d

0 occurrences of 0x1e

0 occurrences of 0x1f

2731 occurrences of 0x20

35492 occurrences of !

0 occurrences of "

0 occurrences of #

0 occurrences of $

0 occurrences of %

0 occurrences of &

0 occurrences of '

0 occurrences of (

0 occurrences of )

0 occurrences of *

0 occurrences of +

0 occurrences of ,

0 occurrences of -

0 occurrences of .

0 occurrences of /

0 occurrences of 0

0 occurrences of 1

0 occurrences of 2

0 occurrences of 3

0 occurrences of 4

0 occurrences of 5

0 occurrences of 6

0 occurrences of 7
```

```
0 occurrences of 8

0 occurrences of 9

0 occurrences of :

0 occurrences of ;

0 occurrences of <

0 occurrences of =

0 occurrences of >

0 occurrences of ?

0 occurrences of @

0 occurrences of A

0 occurrences of B

0 occurrences of C

0 occurrences of D

0 occurrences of E

0 occurrences of F

0 occurrences of G

2732 occurrences of H

0 occurrences of I

0 occurrences of J

0 occurrences of K

0 occurrences of L

0 occurrences of M

0 occurrences of N

0 occurrences of O

0 occurrences of P

0 occurrences of Q

0 occurrences of R

0 occurrences of S

0 occurrences of T

0 occurrences of U
```

```
0 occurrences of V

2731 occurrences of W

0 occurrences of X

0 occurrences of Y

0 occurrences of Z

0 occurrences of [

0 occurrences of \

0 occurrences of ]

0 occurrences of ^

0 occurrences of _

0 occurrences of `

0 occurrences of a

0 occurrences of b

0 occurrences of c

2731 occurrences of d

2732 occurrences of e

0 occurrences of f

0 occurrences of g

0 occurrences of h

0 occurrences of i

0 occurrences of j

0 occurrences of k

8194 occurrences of l

0 occurrences of m

0 occurrences of n

5462 occurrences of o

0 occurrences of p

0 occurrences of q

2731 occurrences of r

0 occurrences of s
```

```
0 occurrences of t

0 occurrences of u

0 occurrences of v

0 occurrences of w

0 occurrences of x

0 occurrences of y

0 occurrences of z

0 occurrences of {

0 occurrences of |

0 occurrences of }

0 occurrences of ~

0 occurrences of 0x7f


C:\Users\czar__000\Desktop\csc415\homework4>
```

The following is the C code of the posix version of the 4 threaded program:

```
//Author: Artem Tolstov

//Purpose: CSC415 HW4

//Function: Takes name of file from command line and counts the number of
ascii characters using

//               predefined number of threads then prints to the screen
occurences of said characters.




#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <pthread.h> //for posix threads

#define BUFFER_SIZE 65536

#define NUM_THREADS 4

void printOccurences(int array[NUM_THREADS][128]);

int charCounts[NUM_THREADS][128];

char buffer[BUFFER_SIZE];

struct threadParams

{

      int threadIndex;

      int stringLengthInBuffer;


};

void *asciiCounter (void *param);


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

{

    //Declarations

    FILE *filePointer;

    int i,j,x;
```

```
      struct threadParams params[NUM_THREADS];
      pthread_t ThreadId;
      pthread_attr_t attr;
//    HANDLE ThreadHandle;



    //Makes sure there are 2 arguments in the command line.
    if (argc != 2 )
      {
            printf( "proper usage: %s filename.extension", argv[0] );
      }
    else
      {
            //Reports if file does not exist.
            filePointer = fopen( argv[1], "r" );

            if (filePointer == 0)
            {
                  printf ("Could not open file\n");
            }
            else
            {
                i = 0;
                  while (((x = fgetc(filePointer)) != EOF) && (i <
BUFFER_SIZE))
                  {
                        //Put next character into i position of buffer,
increment i.
                      buffer[i] = x; i++;
                        //printf( "%c", x);
```

```
                    }

                    fclose( filePointer );


                    //For debugging

                    //printf( "\nbuffer contains: %s\n", buffer );

                    printf("Length of string in file is: %d\n",
strlen(buffer));




                    //Initialize array charCounts to 0

                    for(i = 0; i < NUM_THREADS; i++)

                    {

                          for(j = 0; j < 128; j++)

                          {

                                charCounts[i][j] = 0;

                          }

                    }


                    //Sends params with thread indexi to threads

                    pthread_attr_init(&attr);

                    for(i = 0; i < NUM_THREADS; i++)

                    {

                          params[i].threadIndex = i;

                          params[i].stringLengthInBuffer = strlen(buffer);

                          pthread_create(&ThreadId, &attr, asciiCounter,
&params[i]);

                          //ThreadHandle = CreateThread(NULL, 0, asciiCounter,
&params[i], 0, &ThreadId);


                    }


                    //waits for threads to finish
                    pthread_join(ThreadId, NULL);

                    //if (ThreadHandle != NULL)

                    //{

                    //    WaitForSingleObject(ThreadHandle, INFINITE);

                    //}


                    //CloseHandle(ThreadHandle);

                    //pthread_exit(0);


                    printOccurences(charCounts);

                    pthread_exit(0);

              }

        }





}


//Prints letter if printable, Hex value if unprintable character.

//Combines the values of the different threads into [0][] section of the
array.

void printOccurences(int array[NUM_THREADS][128])

{

    int i,j, holdIt;

    for(i = 0; i < 128; i++)

    {

        //holdIt = array[0][i];

            //array[0][i] = 0;

        for(j = 1; j < NUM_THREADS; j++)

        {

           array[0][i] +=  array[j][i];
        }
        //array[0][i] = array[0][i] + holdIt;
    }


    for(i = 0; i < 128; i++)
    {
        if((i > 32) && (i != 127))
        {
            printf("%d occurrences of %c\n", array[0][i], (char)i);
        }
        else
        {
            printf("%d occurrences of 0x%x\n", array[0][i], i);
        }
    }


}


void *asciiCounter (void *param)
{
      struct threadParams thisThreadsParams = *(struct threadParams*)param;
      int i;
      //printf("asciiCounter has threadIndex = %d\n", 
thisThreadsParams.threadIndex);
      for(i = thisThreadsParams.threadIndex * 
(thisThreadsParams.stringLengthInBuffer/NUM_THREADS); i < 
(thisThreadsParams.threadIndex+1)*(thisThreadsParams.stringLengthInBuffer/NUM
_THREADS); i++)
      {
            //printf("%c is value of %d\n", buffer[i], (int)buffer[i]);
            charCounts[thisThreadsParams.threadIndex][(int)buffer[i]]++;
            //Keeps the null characters in the buffer from writing
      }

      //final check to count remaining characters due to truncation in 
division

      if(thisThreadsParams.threadIndex + 1 == NUM_THREADS)

      {


      if(NUM_THREADS*(thisThreadsParams.stringLengthInBuffer/NUM_THREADS) != 
thisThreadsParams.stringLengthInBuffer)

            {

                  for(i = 
NUM_THREADS*(thisThreadsParams.stringLengthInBuffer/NUM_THREADS); i < 
thisThreadsParams.stringLengthInBuffer; i++)

                  {


      charCounts[thisThreadsParams.threadIndex][(int)buffer[i]]++;

                  }

            }

      }



      return 0;

}
```

The following is the execution of the posix version of the 4 threaded running the same 64kb test.txt file that says “Hello World!!!!!!!!!!!” over and over:

```
liteshotv3@liteshotv3-VirtualBox:~/Desktop/CSC415/hw4$ ./posix4 test.txt
Length of string in file is: 65536
0 occurrences of 0x0
0 occurrences of 0x1
0 occurrences of 0x2
0 occurrences of 0x3
0 occurrences of 0x4
0 occurrences of 0x5
0 occurrences of 0x6
0 occurrences of 0x7
0 occurrences of 0x8
0 occurrences of 0x9
0 occurrences of 0xa
0 occurrences of 0xb
0 occurrences of 0xc
0 occurrences of 0xd
0 occurrences of 0xe
0 occurrences of 0xf
0 occurrences of 0x10
0 occurrences of 0x11
0 occurrences of 0x12
0 occurrences of 0x13
0 occurrences of 0x14
0 occurrences of 0x15
0 occurrences of 0x16
0 occurrences of 0x17
0 occurrences of 0x18
0 occurrences of 0x19
0 occurrences of 0x1a
0 occurrences of 0x1b
0 occurrences of 0x1c
0 occurrences of 0x1d
0 occurrences of 0x1e
0 occurrences of 0x1f
2731 occurrences of 0x20
35492 occurrences of !
0 occurrences of "
0 occurrences of #
0 occurrences of $
0 occurrences of %
0 occurrences of &
0 occurrences of '
0 occurrences of (
0 occurrences of )
0 occurrences of *
0 occurrences of +
0 occurrences of ,
0 occurrences of -
0 occurrences of .
0 occurrences of /
0 occurrences of 0
0 occurrences of 1
0 occurrences of 2
```

```
0 occurrences of 3
0 occurrences of 4
0 occurrences of 5
0 occurrences of 6
0 occurrences of 7
0 occurrences of 8
0 occurrences of 9
0 occurrences of :
0 occurrences of ;
0 occurrences of <
0 occurrences of =
0 occurrences of >
0 occurrences of ?
0 occurrences of @
0 occurrences of A
0 occurrences of B
0 occurrences of C
0 occurrences of D
0 occurrences of E
0 occurrences of F
0 occurrences of G
2732 occurrences of H
0 occurrences of I
0 occurrences of J
0 occurrences of K
0 occurrences of L
0 occurrences of M
0 occurrences of N
0 occurrences of O
0 occurrences of P
0 occurrences of Q
0 occurrences of R
0 occurrences of S
0 occurrences of T
0 occurrences of U
0 occurrences of V
2731 occurrences of W
0 occurrences of X
0 occurrences of Y
0 occurrences of Z
0 occurrences of [
0 occurrences of \
0 occurrences of ]
0 occurrences of ^
0 occurrences of _
0 occurrences of `
0 occurrences of a
0 occurrences of b
0 occurrences of c
2731 occurrences of d
2732 occurrences of e
0 occurrences of f
0 occurrences of g
0 occurrences of h
0 occurrences of i
0 occurrences of j
0 occurrences of k
```

```
8194 occurrences of l
0 occurrences of m
0 occurrences of n
5462 occurrences of o
0 occurrences of p
0 occurrences of q
2731 occurrences of r
0 occurrences of s
0 occurrences of t
0 occurrences of u
0 occurrences of v
0 occurrences of w
0 occurrences of x
0 occurrences of y
0 occurrences of z
0 occurrences of {
0 occurrences of |
0 occurrences of }
0 occurrences of ~
0 occurrences of 0x7f
```