Kurtis Chan ID: 918139175
Github: kchanw CSC415-03 Operating Systems

Assignment 4 – Word Blast

Description:Write a program to read and tally the occurrences of words within a given file.
This is to be done with threads for reading with mutex locks around critical sections.
Once tallying is done, display the top 10 most common words and how many times they occur.
Only use functions like open, pread, read, lseek, close for file.

Approach / What I Did:

First I went to look at the functions mentioned, like lseek, pread, open, and close on the manpage. I did look into read as well, but felt like pread would fit better since I would read at an offset rather than buffering part of the file. In order to store a word and its count, I made a struct called “Word” with a character array of size 30 and an integer count so that it can store a word and its count. Then initialized a pointer of struct Word called wordList to store different words and their counts. Next I initialized listSize which is a volatile int, I did this because the size will change when adding a word, and when adding a word, it should be a critical section since you are changing a global variable(listSize should also be in the critical section). Next the file descriptor and the # of threads which are ints, while the fileSize and the block(portion of file) are off_t since that is what lseek returns when getting how big the file is. Next I allocated memory to wordList of sizeof “struct Word” so that it can store one when you add it. Next I checked whether or not there is a file name and # of threads, if any are NULL then it would pop up a message and exit.
Next would be to get the file descriptor and open the file so I can get the size of the file and divide it up. So I use open in read only mode. Through discord someone mentioned to use the atoi() function to convert the arg[2] which is a char* to an int to get the number of threads. After that I used lseek to get the current(which is 0 at the time) and end of the file which is offset and fileSize. block which is how big a thread will read is given by fileSize/threads. Lastly set the file offset back to the beginning of the file using offset.
Next I need to get the positions where each thread will start to read, so I store the ending position of each one in a pointer to the array of ending positions. Now to make the function to pass into pthread, I start off by making a buffer to store a block of text. To get the starting point I did the arg which is the end position minus the block which is how much text is read. For a single thread the start would be 0, but for a double the start would be 0 for one and 1679774 for the other. Next strtok_r will be stored in token and as long as token is not null, it will keep running. When token is a six characters or more then it will call addWord(token). addWord is supposed add a word to the list if it can’t find it in the list already(reallocate memory so it can hold it aswell) and when it does find it in the list, it will +1 to the count. The critical sections in here are around anything dealing with globals like wordList, listSize, including inside the function strcpy.

To call pthread according to the number of threads, there is a for loop giving each pthread a different bufferposition/offset from the buffposition array. To wait for all of the threads to finish, I use a for loop for pthread_join with their tids.
For the sorting I used a bubblesort, though it is not the most efficient, I felt that this would be easier for me to code compared to selection or quicksort. To print out the top 10, I printed the last 10 elements with their counts and finally freed the wordList and closed the file.

Issues and Resolutions:

The first thing to note is that I see that my output is wrong and when I try to increase the listSize I run into a segmentation fault. Running valgrind it seems that I am stepping into registers I should not both reading and writing. From reading the man page on strcpy and strncpy I am guessing that I am either not allocating enough space for the string to be copied and it is overflowing, or strcpy is going past what I gave it since it doesn't have a '\0' terminating it which results in me being somewhere I shouldn't. Either way through valgrind I see that I am not handling memory correctly and I am accessing places I am not supposed to. I had print statements to show starting and ending positions, you can see that they aren't in order since the threads are working in parallel, so there is no telling which one goes first and next beforehand. **Note: when position is printed out, that is end position**

Analysis:

Though my output is not correct, I can see the differences in the using multiple threads compared to 1 and 2. When going to 4, the time increased and did the same for 8 when comparing to 2, but still faster than 1. My guess to why it doesn't go much faster with more than 2 is because in the virtual box manager says it has 2 processors and when googling how many threads can a core handle, the general answer is 1 given there is no "hyperthreading" which I am not exactly sure of the meaning. From what I looked up it sounds like it lets you take in more instructions which makes it run faster.

Screen shots
With 1

```
student@student-VirtualBox:~/csc415-assignment-4-word-blast-kchanw$ make run RUNOPTIONS="WarAndPeace.txt 1"
./Chan_Kurtis_HW4_main WarAndPeace.txt 1
1
File Size: 3359549
 blocks: 3359549
offset: 0
position: 3359549
starting point: 0
Number 1 is eBooks. with a count of 1
Number 2 is  with a count of 0
Number 3 is  with a count of 0
Number 4 is  with a count of 0
Number 5 is  with a count of 0
Number 6 is  with a count of 0
Number 7 is  with a count of 0
Number 8 is l with a count of 0
Number 9 is  with a count of 73320048
Number 10 is  with a count of 0
Total Time was 0.052853026 seconds
student@student-VirtualBox:~/csc415-assignment-4-word-blast-kchanw$ make run RUNOPTIONS="WarAndPeace.txt 2"
gcc -c -o Chan_Kurtis_HW4_main.o Chan_Kurtis_HW4_main.c -g -I.
Chan_Kurtis_HW4_main.c: In function 'main':
```

With 2

```
gcc -o Chan_Kurtis_HW4_main Chan_Kurtis_HW4_main.o -g -I. -l pthread
./Chan_Kurtis_HW4_main WarAndPeace.txt 2
2
File Size: 3359549
 blocks: 1679774
offset: 0
position: 1679774
position: 3359548
starting point: 1679774
starting point: 0
Number 1 is eBooks. with a count of 1
Number 2 is  with a count of 0
Number 3 is  with a count of 0
Number 4 is  with a count of 0
Number 5 is  with a count of 0
Number 6 is  with a count of 0
Number 7 is  with a count of 0
Number 8 is  with a count of 0
Number 9 is  with a count of 231212656
Number 10 is  with a count of 0
Total Time was 0.045721516 seconds
student@student-VirtualBox:~/csc415-assignment-4-word-blast-kchanw$ make run RUNOPTIONS="WarAndPeace.txt 4"
./Chan_Kurtis_HW4_main WarAndPeace.txt 4
```

With 4

```
Total Time was 0.045721516 seconds
student@student-VirtualBox:~/csc415-assignment-4-word-blast-kchanw$ make run RUNOPTIONS="WarAndPeace.txt 4"
./Chan_Kurtis_HW4_main WarAndPeace.txt 4
4
File Size: 3359549
 blocks: 839887
offset: 0
position: 839887
position: 1679774
position: 2519661
position: 3359548
starting point: 2519661
starting point: 839887
starting point: 0
starting point: 1679774
Number 1 is eBooks. with a count of 1
Number 2 is  with a count of 0
Number 3 is  with a count of 0
Number 4 is  with a count of 0
Number 5 is  with a count of 0
Number 6 is  with a count of 0
Number 7 is  with a count of 0
Number 8 is  with a count of 0
Number 9 is  with a count of 2030323312
Number 10 is  with a count of 0
Total Time was 0.056525305 seconds
student@student-VirtualBox:~/csc415-assignment-4-word-blast-kchanw$ make run RUNOPTIONS="WarAndPeace.txt 8"
```

With 8

```
student@student-VirtualBox:~/csc415-assignment-4-word-blast-kchanw$ make run RUNOPTIONS="WarAndPeace.txt 8"
./Chan_Kurtis_HW4_main WarAndPeace.txt 8
8
File Size: 3359549
 blocks: 419943
offset: 0
position: 419943
position: 839886
position: 1259829
position: 1679772
position: 2099715
position: 2519658
position: 2939601
position: 3359544
starting point: 0
starting point: 2939601
starting point: 1679772
starting point: 1259829
starting point: 839886
starting point: 2099715
starting point: 2519658
starting point: 419943
Number 1 is riding with a count of 1
Number 2 is  with a count of 0
Number 3 is  with a count of 0
Number 4 is  with a count of 0
Number 5 is  with a count of 0
Number 6 is  with a count of 0
Number 7 is  with a count of 0
Number 8 is > with a count of 0
Number 9 is  with a count of -1771088272
Number 10 is  with a count of 0
Total Time was 0.046813916 seconds
student@student-VirtualBox:~/csc415-assignment-4-word-blast-kchanw$
```