Hajime Miyazaki ID: 922402751
Github: hMiyazaki95 CSC415 Operating Systems

Assignment 4 - Word Blast

Description:
To read WarandPeace and count and tally each of the words that are 6 or more characters long by only using linux file functions.

Approach / What I Did:

At the very beginning, I started with understanding the whole idea of this assignment by reading the Readme.md file carefully. Then I divided the coding work into different sections.

First I started writing a main function that takes two arguments. Inside the main function, I declared the char pointer that points to the text file called WarAndPeace. Also I declared an integer variable to count the number of threads. Next, I opened the file read only, saving the result in a file pointer, fd. Then I checked for possible error conditions that could occur in the process of opening the file and while reading the file. If any of the arguments are missing from the command line, and error is thrown. The program expects two arguments, the file name and the number of threads. Then I made sure that the file I'm opening was not empty, as shown by a return value of -1 from lseek. Then if the file is empty, and error is generated. I checked if there are no matches of an integer in the string then error is called. If thread count is less than 1, another error is also generated, I wanted to figure out how long the file length is so I assigned fileLength to the return value of lseek. After that I made sure that lseek succeeds. Then I checked if the file has at least one character. Finally I created an initial range for each thready by dividing the entire length of the file by the number of threads. The main function sets up each thread.

First I called calloc to create an array of each thread object. If calloc succeeds, then it returns a pointer to an array of thread structures that pthread_create requires. I also needed a way to pass multiple argument values to each thread, so I allocated another array large enough to hold the information needed by the threads. I needed to calculate the offset of each chunk of the file from beginning to the end to be processed by each thread. So I created an integer variable bufferOffset and initialized it to 0 and increment it by the size of the file that each thread operates on. Each thread gets one entry in the threadArg array. I didn't want to write the error condition every time, so I created safeMalloc() to prevent code repetition before I initialized thisThread for the memory allocation in the array.

To create multiple threads and process the file in parallel, I used the for loop with counter variable to iterate through the threads. The thread must contain the number of thread start and end of the buffer offset and file descriptor so I added them in the arguments of the thread. Next I used the pthread_create function that takes four arguments to create a new thread. If the return value of pthread_create thread is not equal to 0, this means that the new thread was not successfully created. After I created all threads successfully, I

needed to wait until all of the threads were finished so I used a for loop and pthread_join function. Then I deallocated memory for the thread arguments and the entire thread array using the free() function. Then I closed the file descriptor using close function.

Third, I needed to remember all the words that are 6 letters or more. I choose a hashtable that is an efficient way to do it. In the hashtable, first I needed to store the input words and count of the words so I created the structure that contains the word and count. The word is a char pointer to allocated C string. Then I created two integer variables; hashSize is the size of the hash table; hashMaxCount is keeps track of the maximum count that is found in the hashtable. The hashIndex function returns the index into the hash array where a word structure should be stored. I created an integer variable len and assigned the length of the C string by using strlen function. Then I also created the long variable sum and set it to 0. Then, I used the for loop to iterate over each character to get a C string to generate a sum of all characters in the word. Then I limit that sum to the hashSize so that the index falls within the allocated hash table.

In this hashtable, I choose linear probing instead of using a traditional linked list because it doesn't require additional storage as long as the hashtable is large enough. First I wanted to insert a new word into the hash table so I created the hashInsert function that takes two arguments; one is integer for the thread number used in debugging, the other one the word which is a char pointer to C string. First I used the hashIndex to initialize indexBucket for the input word. After that, I used mutex lock since two threads should not change a global data structure at the same time. Next, since I wanted to find an empty hash bucket or matching bucket, I used linear probing. To do this, I used the while loop to check if the current hashBucket is empty. If not empty it increments the index of the hashBucket and keeps checking the next hash bucket, until we find an empty has bucket. Then, I checked if the hashbucket exceeds the hashtable size. If it exceeds, then it resets the position of the counter to the beginning of the hash table. If it's empty it fills in the word into the bucket and sets the count to 1. Otherwise it increments count since that word is already in the hashBucket. If the word is already in the hashtable, we don't need to keep that word, so we free it. I update the maximum hash count if the new count is now larger.

I needed an empty hash table so I created a hash Create function that takes one argument. Inside the function, I allocated memory for an array of the structure with the size of the array that is equal to the number of buckets. Then, I initialized the word to Null which means that the hash table is currently empty.

Finally, I needed to free the memory that was allocated in the hash table so I created a hashFree function. Inside the function. I used for loops to iterate through each item of the hashtable and check if the word section in the struct is not empty. If not empty then use the free function to free the allocated memory to avoid the memory leaks. Then set the hash table pointer to Null and hashsize to 0 for safety reasons.

Hajime Miyazaki ID: 922402751
Github: hMiyazaki95 CSC415 Operating Systems

For the sorting part of this assignment, I wanted to sort words by count so I used the doubly linked list to keep track of only the top 10. I will get rid of a word count when it's no longer in the top ten. Then I reused that 10th structure for the new item. After that, I insert a word count into the linked list based on count order, from largest to smallest. First, I created a struct with the prev and next pointer. Next, I wanted to print the top 10 words in the hash table from largest to smallest so I created the hashPrintByCount function that implements an insertion sort style algorithm. To create insertion sort, first I created head pointer and last pointer that points to Null. Then, initialized the itemsInLinkedList to 0. I used a for loop to iterate through each item of the hashtable. Then, check if I needed to insert words and count into the top 10. If it is, I find where to insert it. Next Insert a new item between the previous pointer and this pointer. After that I checked if there is another item after the current item. If not then the current item is the last item. Otherwise, new is the only item in the linkedlist. Next, I handle the case that table has 10 items and I need to insert an item into the table while flushing the last item. To do this, I need to find where to insert a new item based on its count and then reused the last data structure to insert the new item into a doubly linked list. I unlinked the last item from the doubly linked list and reused memory to store the new item. After that, I made a new item that is inserted between the previous and next items. Then, check if there is no next item, it replaces the last item. Then, the new item becomes the last item. Next, I wanted to print and free the items stored in a doubly linked list of struct in order from largest to smallest so I started with initializing a pointer to the head of the doubly linked list and a counter for the number of items printed. Then use the while loop to move the pointer to the next item until it is null. Inside the loop, it prints the word and count of the current item that saves a pointer to the next item in the list. This frees the current item while printing and then moves the pointer to the next item.

I needed a function to move to the end of a word in case a thread is given a buffer that points to the middle of a word. Also if the thread buffer end also points in the middle of a word, it is a most moved to the end of a word. To do this, I created a moveToEndOfWord function that takes two arguments. This function reads one character at a time from the beginning of the specified offset in the text file and checks if the character is a delimiter. If not then the offset is incremented until it finds a delimiter.

Next, I wanted to count the 6 or greater letter words in a subset of the file so I created a threadFunc function that takes one argument. The arguments include the start and end position of the thread's section of the text file. This also includes the file descriptor. In the function, I move the beginning pointer to the end of the word and also move the end pointer to the end of a word. The thread reads the file character by character and finds the beginning and end of each word. Then, I checked if the word is 6 letters or longer. If it is then add the word is added to the hashtable. If there is any error while reading the file, and error is generated. At the end of the function, the thread ignores any consecutive delimiters until it reaches the end of the subset of the file. Finally, I called the hashFree function at the end to clean up the process of hashtable.

Hajime Miyazaki
Github: hMiyazaki95

ID: 922402751
CSC415 Operating Systems

Analysis of the results:
Compared to 4 different runs, there was not a big difference in total runtime on each runs.

Screenshot of compilation:

Runtime
Storage space required.

```
student@student-VirtualBox:~/Documents/csc415-assignment-4-word-blast-hMiyazaki95$ vim miyazaki_hajime_HW4_main.c
student@student-VirtualBox:~/Documents/csc415-assignment-4-word-blast-hMiyazaki95$ make assignment
gcc -c -o miyazaki_hajime_HW4_main.o miyazaki_hajime_HW4_main.c -g -I.
gcc -o miyazaki_hajime_HW4_main miyazaki_hajime_HW4_main.o -g -I. -l pthread
./miyazaki_hajime_HW4_main WarAndPeace.txt 1 > run-1.log
./miyazaki_hajime_HW4_main WarAndPeace.txt 2 > run-2.log
./miyazaki_hajime_HW4_main WarAndPeace.txt 4 > run-4.log
./miyazaki_hajime_HW4_main WarAndPeace.txt 8 > run-8.log
./miyazaki_hajime_HW4_main WarAndPeace.txt 16 > run-16.log
valgrind ./miyazaki_hajime_HW4_main  2 >vrun.log 2>&1
Makefile:234: recipe for target 'vrun.log' failed
make: [vrun.log] Error 1 (ignored)
./checkLog < run-1.log > check-1.log
diff check-1.log run-1.log
10a11
> Total Time was 4.122703207 seconds
Makefile:133: recipe for target 'verify-1.log' failed
make: [verify-1.log] Error 1 (ignored)
************* Expect error when checking error conditions
./miyazaki_hajime_HW4_main
./miyazaki_hajime_HW4_main: 2 arguments are required: filename and threadcount
Makefile:238: recipe for target 'error-test' failed
make: [error-test] Error 1 (ignored)
./miyazaki_hajime_HW4_main notEnoughArguments
./miyazaki_hajime_HW4_main: 2 arguments are required: filename and threadcount
Makefile:238: recipe for target 'error-test' failed
make: [error-test] Error 1 (ignored)
./miyazaki_hajime_HW4_main fileDoesNotExist 1
Could not open fileName 'fileDoesNotExist' No such file or directory
Makefile:238: recipe for target 'error-test' failed
make: [error-test] Error 2 (ignored)
./miyazaki_hajime_HW4_main WarAndPeace.txt xxxxx
./miyazaki_hajime_HW4_main: threadCount argument requires an integer
Makefile:238: recipe for target 'error-test' failed
make: [error-test] Error 3 (ignored)
./miyazaki_hajime_HW4_main WarAndPeace.txt -2
./miyazaki_hajime_HW4_main: threadCount must be greater than 0

Makefile:238: recipe for target 'error-test' failed
```

Hajime Miyazaki
Github: hMiyazaki95

ID: 922402751
CSC415 Operating Systems

```
student@student-VirtualBox:~/Documents/csc415-assignment-4-word-blast-hMiyazaki9
5$ make run
./miyazaki_hajime_HW4_main WarAndPeace.txt 2
Number 1 is Pierre with a count of 1963
Number 2 is Prince with a count of 1577
Number 3 is Natásha with a count of 1213
Number 4 is Andrew with a count of 1143
Number 5 is himself with a count of 1017
Number 6 is French with a count of 881
Number 7 is before with a count of 779
Number 8 is Rostóv with a count of 776
Number 9 is thought with a count of 766
Number 10 is CHAPTER with a count of 730
Total Time was 5.334509813 seconds
student@student-VirtualBox:~/Documents/csc415-assignment-4-word-blast-hMiyazaki9
5$ make run
```

```
student@student-VirtualBox:~/Documents/csc415-assignment-4-word-blast-hMiyazaki9
5$ make run
./miyazaki_hajime_HW4_main WarAndPeace.txt 2
Number 1 is Pierre with a count of 1963
Number 2 is Prince with a count of 1577
Number 3 is Natásha with a count of 1213
Number 4 is Andrew with a count of 1143
Number 5 is himself with a count of 1017
Number 6 is French with a count of 881
Number 7 is before with a count of 779
Number 8 is Rostóv with a count of 776
Number 9 is thought with a count of 766
Number 10 is CHAPTER with a count of 730
Total Time was 5.242379472 seconds
student@student-VirtualBox:~/Documents/csc415-assignment-4-word-blast-hMiyazaki9
5$ make run
```

Hajime Miyazaki
Github: hMiyazaki95

ID: 922402751
CSC415 Operating Systems

```
student@student-VirtualBox:~/Documents/csc415-assignment-4-word-blast-hMiyazaki9
5$ make run
./miyazaki_hajime_HW4_main WarAndPeace.txt 2
Number 1 is Pierre with a count of 1963
Number 2 is Prince with a count of 1577
Number 3 is Natásha with a count of 1213
Number 4 is Andrew with a count of 1143
Number 5 is himself with a count of 1017
Number 6 is French with a count of 881
Number 7 is before with a count of 779
Number 8 is Rostóv with a count of 776
Number 9 is thought with a count of 766
Number 10 is CHAPTER with a count of 730
Total Time was 5.258991588 seconds
student@student-VirtualBox:~/Documents/csc415-assignment-4-word-blast-hMiyazaki9
5$ make run
```

```
student@student-VirtualBox:~/Documents/csc415-assignment-4-word-blast-hMiyazaki9
5$ make run
./miyazaki_hajime_HW4_main WarAndPeace.txt 2
Number 1 is Pierre with a count of 1963
Number 2 is Prince with a count of 1577
Number 3 is Natásha with a count of 1213
Number 4 is Andrew with a count of 1143
Number 5 is himself with a count of 1017
Number 6 is French with a count of 881
Number 7 is before with a count of 779
Number 8 is Rostóv with a count of 776
Number 9 is thought with a count of 766
Number 10 is CHAPTER with a count of 730
Total Time was 5.312916621 seconds
student@student-VirtualBox:~/Documents/csc415-assignment-4-word-blast-hMiyazaki9
5$
```