Sukrit Dev Dhawan ID: 922432027
Github: SukritD27 CSC415 Operating Systems

Assignment 4 – Word Blast

Description:
This assignment is to write a C program that reads from a file given in the command line argument and goes through the file and checks the count of words with length 6 or more and adds them into a custom data structure. This program also implements multiple threads so that it becomes faster.

Approach / What I Did:
I started out by making the custom data structure because I knew nothing about threading in the start. I started out by trying to build a hashmap and then having the key as unique words and values as the count of how many times they appear. This method could be viable but it does not make sense if i need to sort it at the end which means going through all the hashmap and then sorting it which would make it as slow as just an array. So I went with the idea of an array of structs and the said structs would contain a word and a count of that word.

My program starts with saving command line arguments for taking the path and the number of threads we want to split our program into. Then it goes into opening the file with the path taken from the command line. I read that file into my buffer of size taken from lseek(). I then create my array of structs and give it an initial size. I then create my threads and give them my fileParser method and my array as a parameter.

My fileParser() method breaks the above said buffer into the number of threads provided. I do that by memcpying part of the buffer into a new buffer. I have a startingPoint pointer that first points to the start of my global buffer. I increment this pointer inside a mutex lock so that each thread can start copying their buffer from their point in the buffer. This is essential for my code to run so that all the threads do not get the same buffer. To check if this was working correctly, I also had a test.txt file with random words.

Then I start tokenizing the buffer in each thread and start adding words into my array of structs. To add a new item in the array, I make a new struct object and add the word and and a count of 1. To check the cases for duplicates, I have a for loop and it goes through the partially filled array and if there exists a word same as the one we are trying to add, it increases its count and breaks out of the for loop. I also have a boolean variable to check for duplicates. If it is true, the while loop will continue and skip the adding of a new struct object in the array.

I also have an incrementer to check the total number of elements filled in the array. I use that to check if the array is filled or not. If it is filled, I just reallocate the array into a new memory location and assign it a new size which is double of the previous one. This is also used for adding new structs into my array of structs.

Then to sort the array, I implemented quick sort so that it is fast. This is done in the main method after the thread have finished. After this i print out the top ten from the sorted array.

Issues and Resolutions:

In terms of issues, I had tons:

1. First one was, I had an insert function and that function helped me add stuff into my array of structs. But this gave me weird results with wrong counts of words. I could not figure it out so i started to make it again and just include it into the thread function. It turns out the only problem was that when I looked for duplicates, I did not hve a way to stop it from adding a new struct to the array of structs. That's why the count was wrong.

2. I got multiple segmentation faults from mallocing my array. I then tried calloc() and it stopped. I think this was because malloc gives random memory and calloc gives continuous memory. Going through all those memory addresses caused it to point at some random memory access by mistake and caused the segmentation fault.

3. Just when I had a ray of hope that the program was running, I saw the count getting messed up again. I had no idea why this was happening. Because of this I was alos getting more segmentation faults. But I figured it out later that my number of elements in the array was getting incremented in a separate mutex lock separate from the lock that protected the adding into the array. This caused that count to mess up.

4. Another thing with the same increment as mentioned above, there was a problem because the count was still bad. I was getting a count of different unique words up like a 1000. This was happening because of me making that counter 0 for each thread at the start and messing up the count. As soon as I took that statement out, everything was fixed and my program finally started working.

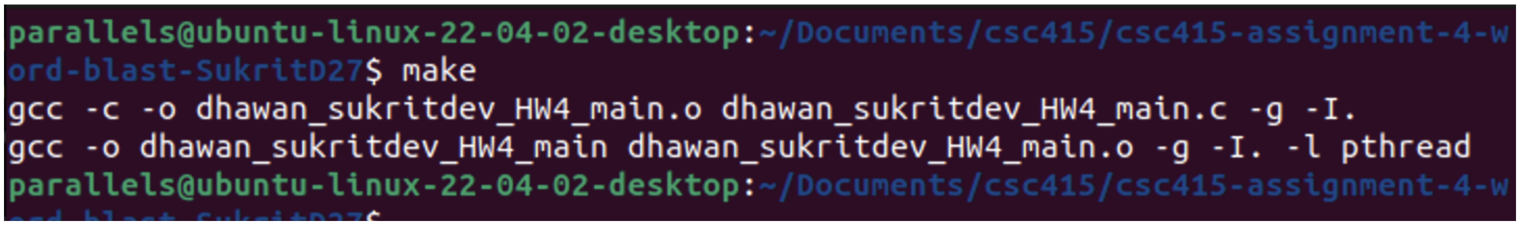
5. I also incremented the same incrementer accidentally when I added a duplicate into that array. This made the count get messed up once more and debugging this one statement took me 2 hours.

6. When I checked for the length of the token I forgot to get the next token. Instead, I just said continue the loop which made it go in an infinite loop. Same thing for if I found any duplicates. I forgot to get the next token which gave me a segmentation fault because it kept changing the count of the same word forever non stop.

Analysis: (If required for the assignment)
When we ran the program with just 1 thread we see the time it took for the process to complete is ~1.7 seconds. When we ran the program with 2 threads, we see definite improvement. The time in this case to runt the program was ~0.932 seconds. This is almost half the time it took with 1 thread which was expected because with 2 threads, the work is divided in half. Now when we try the program with 4 threads we do not see a significant improvement in time. It now just took ~0.84 seconds which is only approximately 1 second faster than if we ran the program with 2 threads. This shows that the doubling the threads actually did not in fact half the time in this case. We see a similar case if we jump to 8 threads (~0.845 seconds) which is again not an improvement and is almost similar to the time taken by 4 threads.

These results could be because the virtual machines are only given 2 cores from out computers CPU. Because of only 2 cores, we are working with 1 thread with one core. So when we go more than 2 threads, we do not have enough cores to manage the requested threads, so it ends up working with 2 threads only. That is why for the threads 2, 4, and 8 the time becomes almost the same and similar to that of 2 threads (~0.8 - 0.9 seconds).

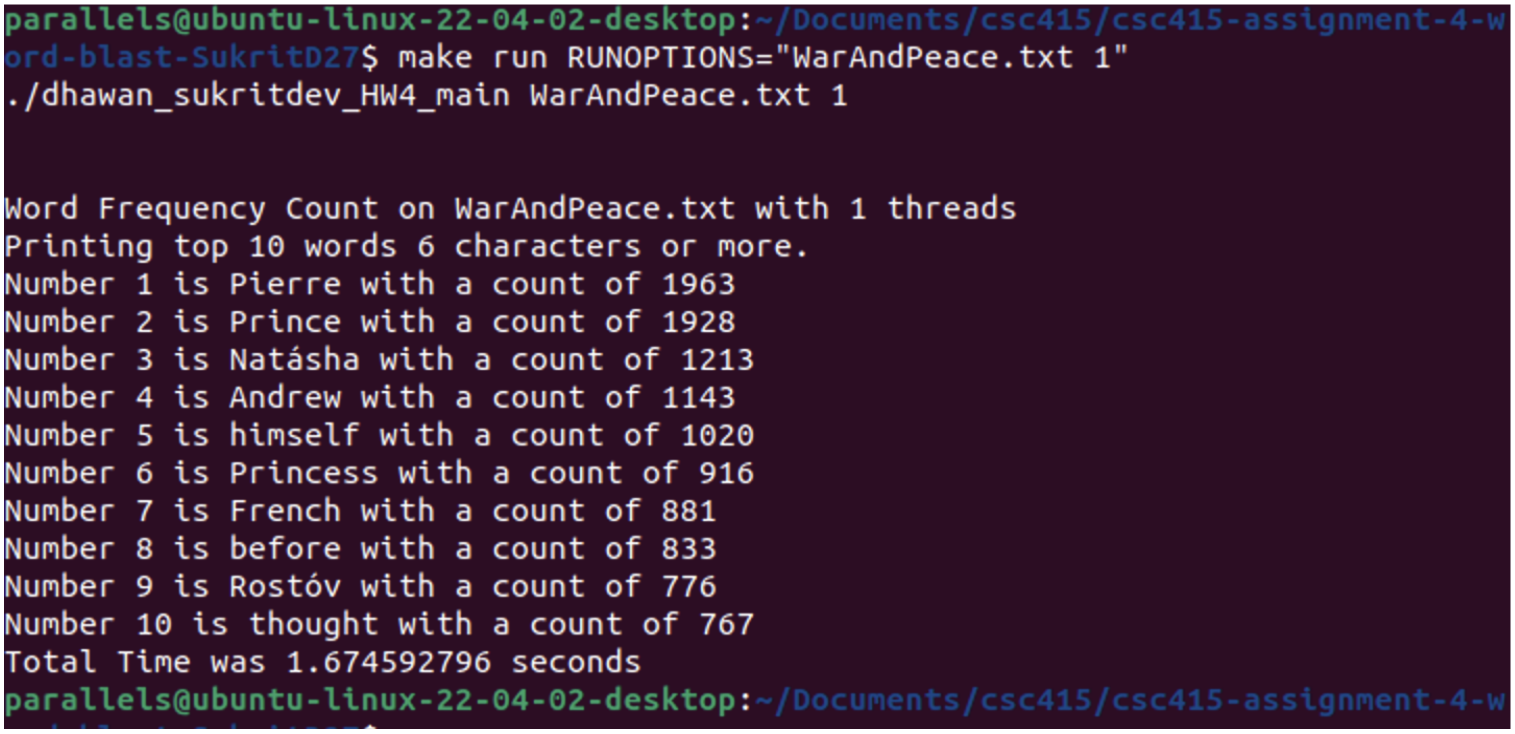
Screen shot of compilation:



```
parallels@ubuntu-linux-22-04-02-desktop:~/Documents/csc415/csc415-assignment-4-w
ord-blast-SukritD27$ make
gcc -c -o dhawan_sukritdev_HW4_main.o dhawan_sukritdev_HW4_main.c -g -I.
gcc -o dhawan_sukritdev_HW4_main dhawan_sukritdev_HW4_main.o -g -I. -l pthread
parallels@ubuntu-linux-22-04-02-desktop:~/Documents/csc415/csc415-assignment-4-w
```

Screen shot(s) of the execution of the program:

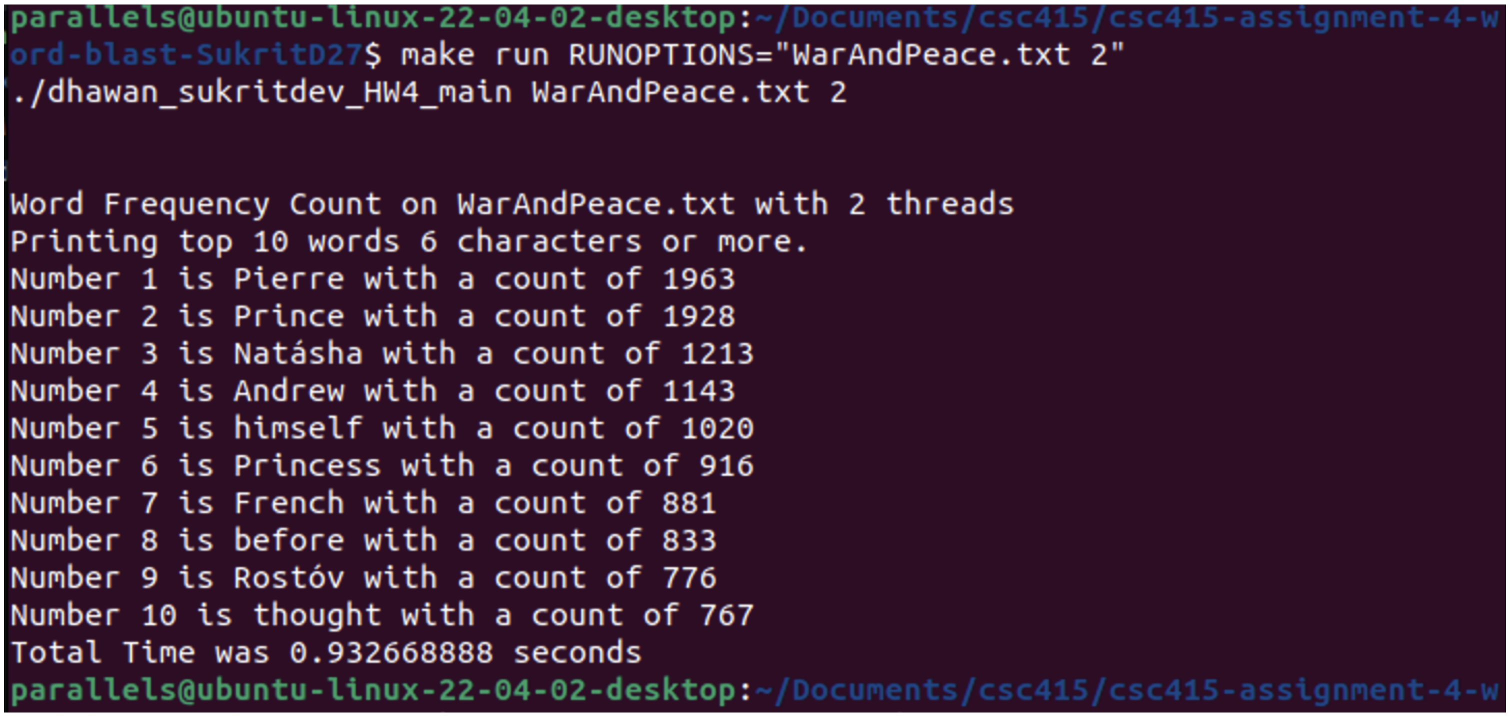
Running the program with **1 thread**:



```
parallels@ubuntu-linux-22-04-02-desktop:~/Documents/csc415/csc415-assignment-4-w
ord-blast-SukritD27$ make run RUNOPTIONS="WarAndPeace.txt 1"
./dhawan_sukritdev_HW4_main WarAndPeace.txt 1


Word Frequency Count on WarAndPeace.txt with 1 threads
Printing top 10 words 6 characters or more.
Number 1 is Pierre with a count of 1963
Number 2 is Prince with a count of 1928
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 1020
Number 6 is Princess with a count of 916
Number 7 is French with a count of 881
Number 8 is before with a count of 833
Number 9 is Rostóv with a count of 776
Number 10 is thought with a count of 767
Total Time was 1.674592796 seconds
parallels@ubuntu-linux-22-04-02-desktop:~/Documents/csc415/csc415-assignment-4-w
```

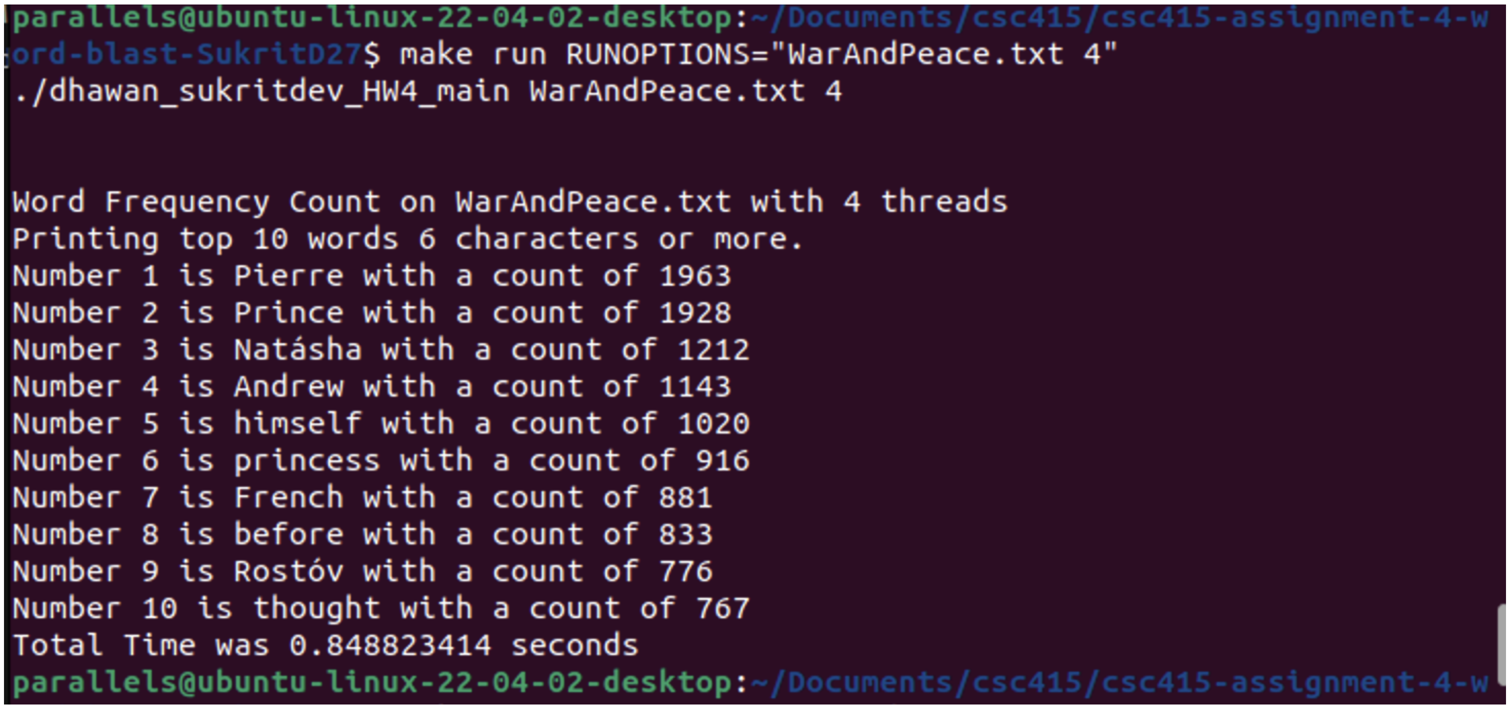
Running the program with **2 threads**:



```
parallels@ubuntu-linux-22-04-02-desktop:~/Documents/csc415/csc415-assignment-4-w
ord-blast-SukritD27$ make run RUNOPTIONS="WarAndPeace.txt 2"
./dhawan_sukritdev_HW4_main WarAndPeace.txt 2


Word Frequency Count on WarAndPeace.txt with 2 threads
Printing top 10 words 6 characters or more.
Number 1 is Pierre with a count of 1963
Number 2 is Prince with a count of 1928
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 1020
Number 6 is Princess with a count of 916
Number 7 is French with a count of 881
Number 8 is before with a count of 833
Number 9 is Rostóv with a count of 776
Number 10 is thought with a count of 767
Total Time was 0.932668888 seconds
parallels@ubuntu-linux-22-04-02-desktop:~/Documents/csc415/csc415-assignment-4-w
```

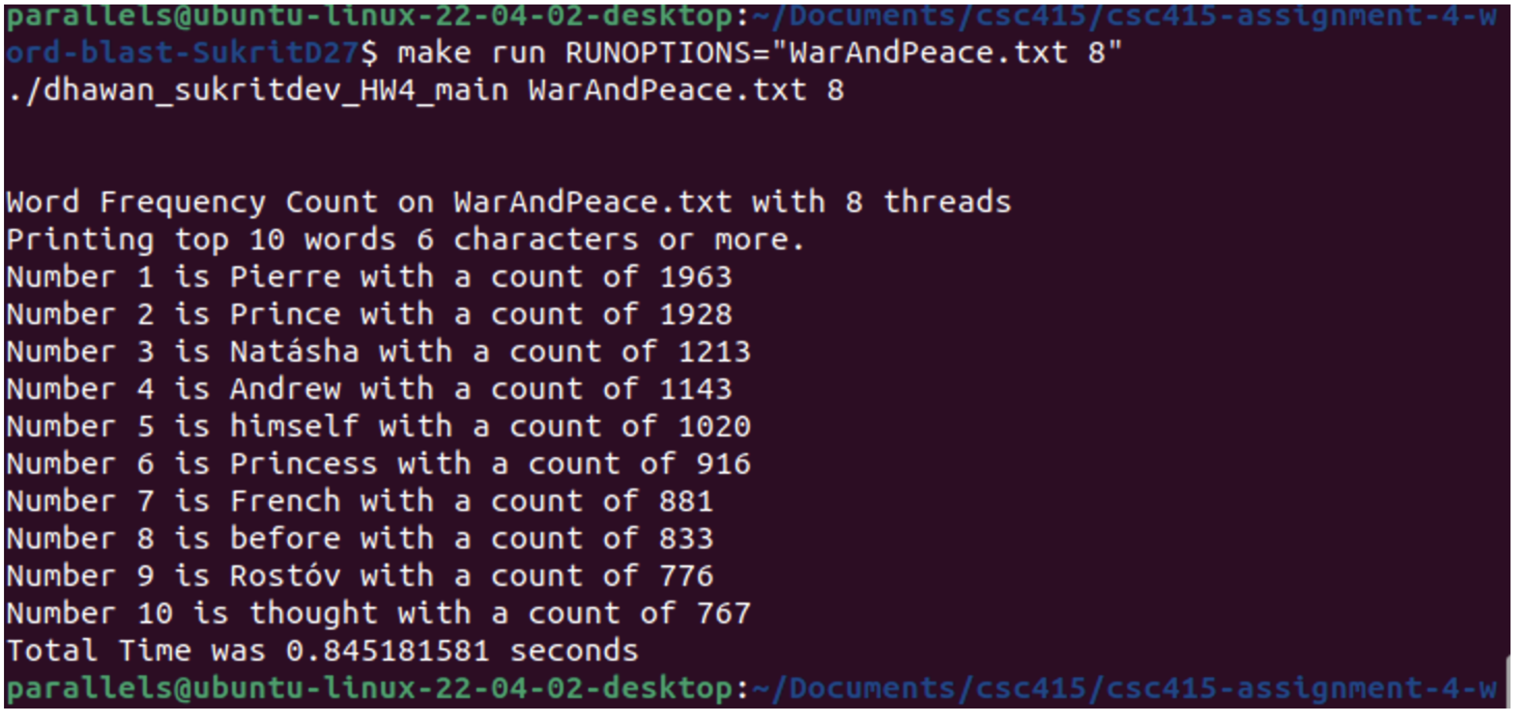
Running the program with **4 threads**:



```
parallels@ubuntu-linux-22-04-02-desktop:~/Documents/csc415/csc415-assignment-4-w
ord-blast-SukritD27$ make run RUNOPTIONS="WarAndPeace.txt 4"
./dhawan_sukritdev_HW4_main WarAndPeace.txt 4


Word Frequency Count on WarAndPeace.txt with 4 threads
Printing top 10 words 6 characters or more.
Number 1 is Pierre with a count of 1963
Number 2 is Prince with a count of 1928
Number 3 is Natásha with a count of 1212
Number 4 is Andrew with a count of 1143
Number 5 is himself with a count of 1020
Number 6 is princess with a count of 916
Number 7 is French with a count of 881
Number 8 is before with a count of 833
Number 9 is Rostóv with a count of 776
Number 10 is thought with a count of 767
Total Time was 0.848823414 seconds
parallels@ubuntu-linux-22-04-02-desktop:~/Documents/csc415/csc415-assignment-4-w
```

Running the program with **8 threads**:



```
parallels@ubuntu-linux-22-04-02-desktop:~/Documents/csc415/csc415-assignment-4-w
ord-blast-SukritD27$ make run RUNOPTIONS="WarAndPeace.txt 8"
./dhawan_sukritdev_HW4_main WarAndPeace.txt 8


Word Frequency Count on WarAndPeace.txt with 8 threads
Printing top 10 words 6 characters or more.
Number 1 is Pierre with a count of 1963
Number 2 is Prince with a count of 1928
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 1020
Number 6 is Princess with a count of 916
Number 7 is French with a count of 881
Number 8 is before with a count of 833
Number 9 is Rostóv with a count of 776
Number 10 is thought with a count of 767
Total Time was 0.845181581 seconds
parallels@ubuntu-linux-22-04-02-desktop:~/Documents/csc415/csc415-assignment-4-w
```