

Assignment 4 – Word Blast

Description:

This assignment is to write a program that will read a text file, parse it, and create a vocabulary of words contained within. This vocabulary contains the word as well as the number of instances of that word that the text file contains. Most importantly, this program is to implement the count in a multi-threaded fashion with the number of threads as an argument to the program. Once the vocabulary is built, the program then outputs the top ten words by count that have a length of 6 or greater. Each run of the program gives a total run time which can be used to evaluate the speed of each run with respect to the number of threads being used.

Approach / What I Did:

I began with planning out what data structure I wanted to use to contain the vocabulary. I knew that I needed a dynamically sized array, so I defined a struct that contained a pointer to the word (char array) and a frequency count of that word. In addition, I created an array struct to contain the word array, its maximum size, and the amount of the size used. I then created functions that initialized an array, inserted new elements into the array, and freed all the array elements (along with the array itself) from memory.

After testing the array, I worked on opening the file and reading its contents into a buffer. After this was completed, I began reading about what the `pthread_create()` function required as arguments. This is when I realized that I needed a structure to contain all the information that each thread needed. I began writing the function that would process each chunk of the text. While doing this, I wrote the thread information struct that would be passed into the function by `pthread_create()`.

Once the chunk processing function was complete, I began assembling the thread information array with the information that each thread would need and called the `pthread_create()` function with its necessary parameters followed by putting together the loop to join all the threads. It was at this point that I began thinking about what portions of the code required mutex locks (critical sections).

At this point, it was time to test the code. I began with long words so that I could better see the program's behavior, and then slowly began decreasing the size of the words. Finally, I implemented a selection sort for the array to sort the array in descending order by count to be printed to the console.

Issues and Resolutions:

I seriously lost count of the issues I ran into. The vast majority of the issues were segmentation faults due to incorrect allocation of memory. After a visit to Prof Bierman's office hours, some of the segmentation errors were corrected. I have to admit, however, that most of the errors were fixed from many hours of blindly stumbling upon them through trial and error. Some of the segmentation faults were fixed by using `calloc()` vs `malloc()`; I understand this is not the ideal solution to this problem and that I should not be having null pointers.

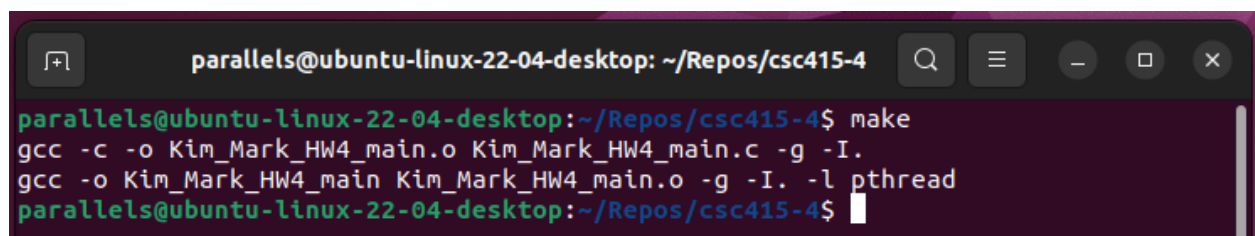
I also ran into some null pointer errors, but resolved them by including an 'if' statement to ignore them. I know this isn't the correct way of resolving this, but I have other classes to attend to and this course has been monopolizing most of my time, so I will have to be okay with "it works—sort of."

Another error I ran into was my data disappearing on me due to assigning the token pointer directly to another pointer. This was fixed after I realized that, of course, if I assign the pointer, it will be pointing to whatever the token pointer was pointing to at the end of the loops, giving me an array of just that one item. I just needed to do a `strcpy()` to copy the contents to a new persistent address.

The last errors I had to fix were duplicate free's of malloc'd pointers. I started by thinking I was smart by creating free functions that would free all the pointers, but was disabused of that notion. This problem was fixed by using free sparingly.

Analysis: When running the program, there is a significant reduction of time when moving from one thread to two (5.67 secs to 3.28 secs). The next two runs of four and eight threads did not appreciably change the run time (3.29 secs and 3.35 secs respectively). This can be accounted for because the virtual machine is only utilizing two CPU cores, which means that its ability to multithread is severely hampered. In addition, running more cores may create thread and process management overhead. Nevertheless, it is the lack of CPU cores available that is preventing an increase of performance beyond two threads.

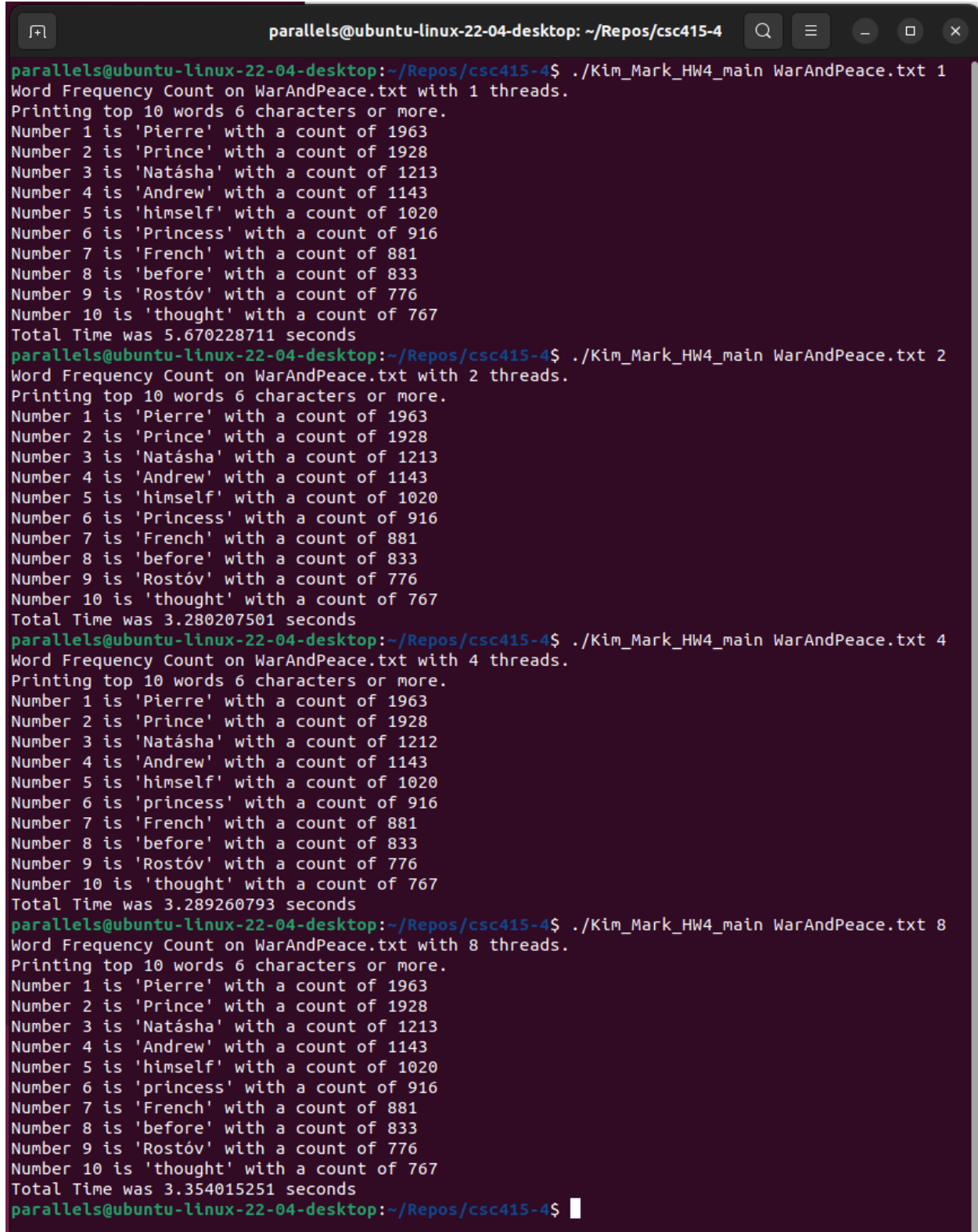
Screen shot of compilation:

A screenshot of a terminal window with a dark background. The window title is "parallels@ubuntu-linux-22-04-desktop: ~/Repos/csc415-4". The terminal shows the following commands and output:

```
parallels@ubuntu-linux-22-04-desktop:~/Repos/csc415-4$ make
gcc -c -o Kim_Mark_HW4_main.o Kim_Mark_HW4_main.c -g -I.
gcc -o Kim_Mark_HW4_main Kim_Mark_HW4_main.o -g -I. -l pthread
parallels@ubuntu-linux-22-04-desktop:~/Repos/csc415-4$
```

SCREEN SHOT OF EXECUTION ON NEXT PAGE

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



```
parallels@ubuntu-linux-22-04-desktop: ~/Repos/csc415-4
parallels@ubuntu-linux-22-04-desktop:~/Repos/csc415-4$ ./Kim_Mark_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 'Natasha' 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 'Rostov' with a count of 776
Number 10 is 'thought' with a count of 767
Total Time was 5.670228711 seconds
parallels@ubuntu-linux-22-04-desktop:~/Repos/csc415-4$ ./Kim_Mark_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 'Natasha' 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 'Rostov' with a count of 776
Number 10 is 'thought' with a count of 767
Total Time was 3.280207501 seconds
parallels@ubuntu-linux-22-04-desktop:~/Repos/csc415-4$ ./Kim_Mark_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 'Natasha' 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 'Rostov' with a count of 776
Number 10 is 'thought' with a count of 767
Total Time was 3.289260793 seconds
parallels@ubuntu-linux-22-04-desktop:~/Repos/csc415-4$ ./Kim_Mark_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 'Natasha' 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 'Rostov' with a count of 776
Number 10 is 'thought' with a count of 767
Total Time was 3.354015251 seconds
parallels@ubuntu-linux-22-04-desktop:~/Repos/csc415-4$
```