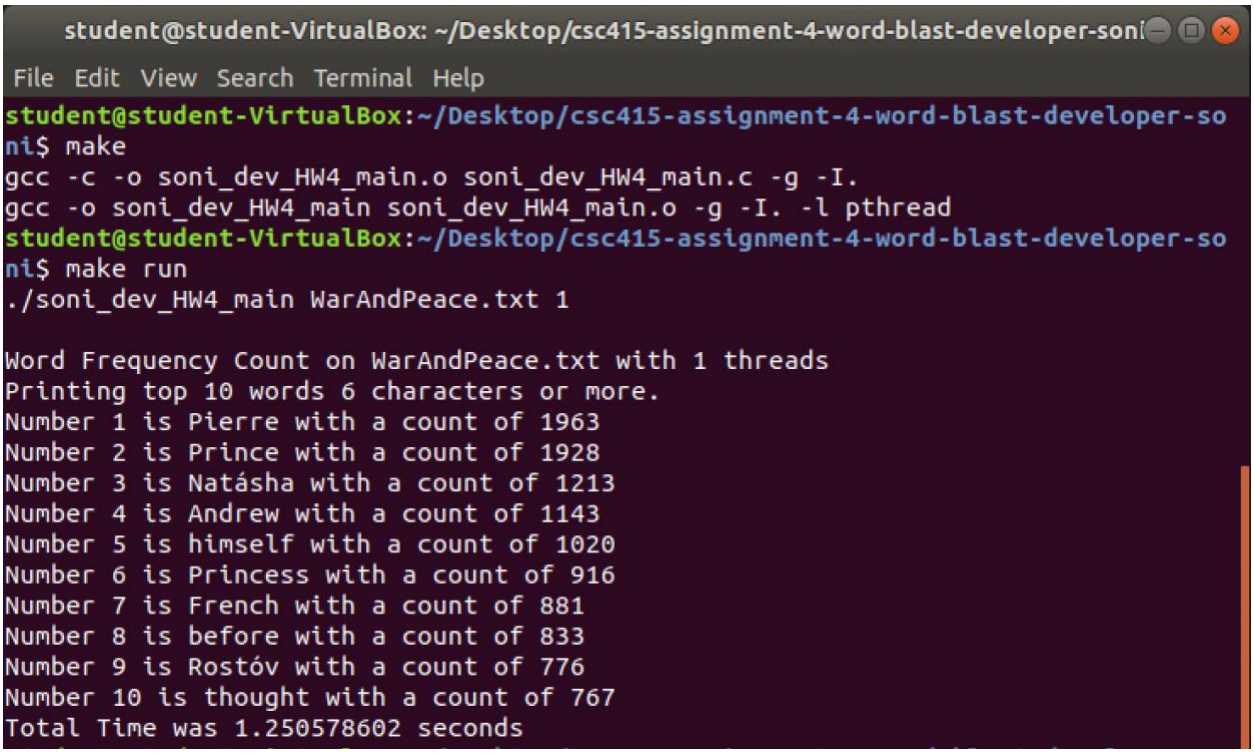
```
/*************************************************************
* Project: Assignment 4 – Word Blast
* File: WriteUp4.pdf
*
* Description: This is the write up file for the program and it includes a
* description of the project, what I did, issues I had, resolutions for the
* issues, reason for different times on different thread runs, compilation
* and execution output from the original program.
*************************************************************/
```

Description: In this program we read a .txt file (WarandPeace.txt, in this case) and count and tally each of the words that are 6 or more characters long. Also made sure that only Linux file functions are used such as open, close, read, lseek and not library functions. We do this process by using threads and each thread takes a chunk of the file and processes it. It then returns the results to the main and main tallies and prints the top 10, 6 or more-character words with the highest frequencies and their tallies. In the main, I used lseek to get the file size and then divided the file size into equal chunks. I create threads and pass each of the chunks into their respective blocks and then call the pthread_create. The adding of the words is done inside a mutex lock. I made a data structure of arrays and named it Pair. This is implemented into a function addWord. This function checks if a match is found and increments the counter by 1 and if its not found then it adds it to the words array. After it is done, the top 10 words are printed each with their respective frequencies in descending order. The total time taken to perform the threading is also displayed at the end. After the process completes, cleanup is also performed to free all the memory for further usage. There were also a number of issues I had during this assignment. The most difficult was to create a data structure and assign all the words in the array and also to count each of their frequency. The latter was easy as we discussed in the class such as creating threads, waiting for them, destroying them and using mutex locks.

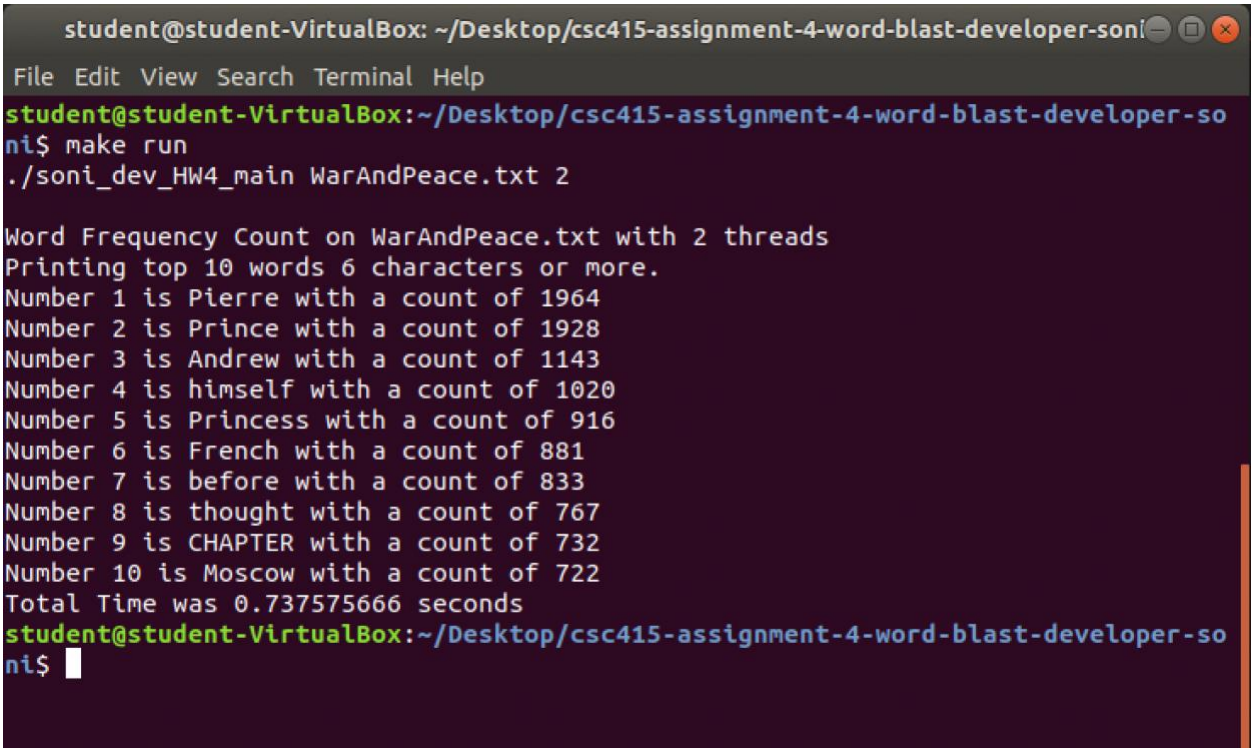
The reason for different times on different thread runs is that as the number of threads increases, the time decreases because of parallelism. The program divides the file into number of threads and performs the operations in parallel. Also, if there are more cores on the operating system, then these threads also use that thus decreasing the time taken for the program to execute.

Compilation and Execution using 1, 2, 4 and 8 threads respectively:

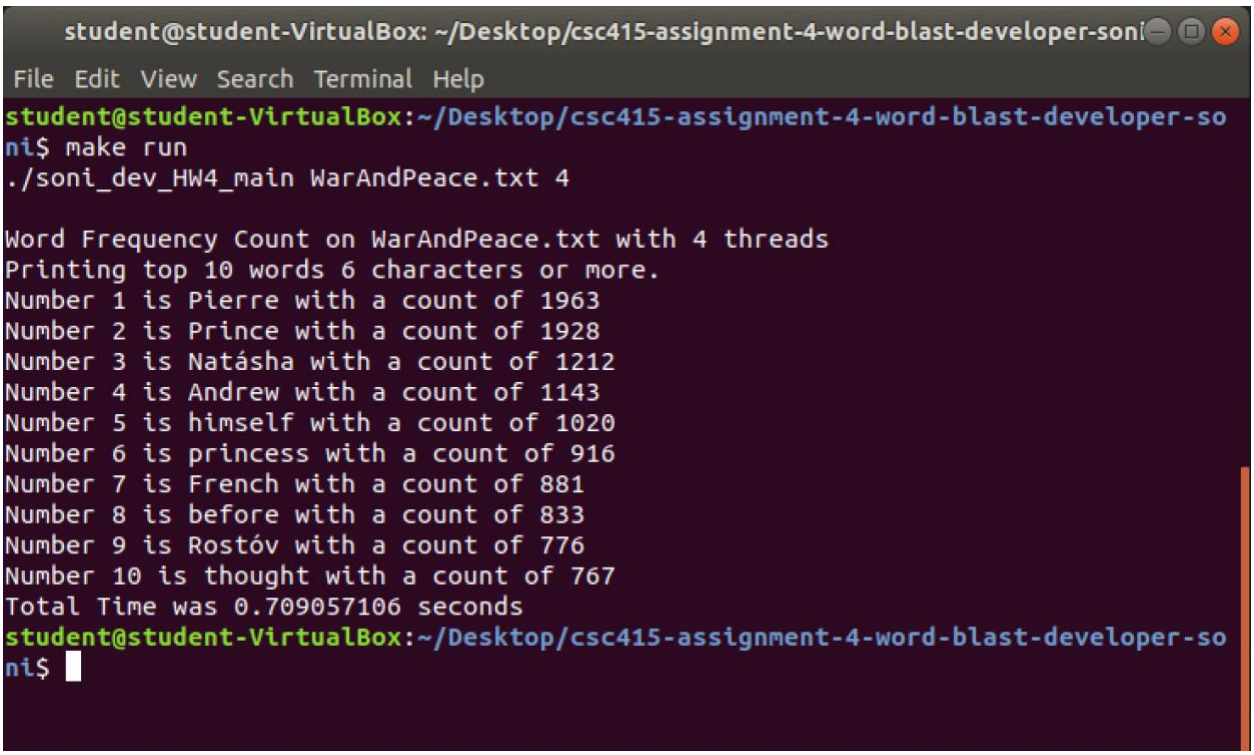
1 Thread :



2 Threads :



4 Threads :



8 Threads :