Juan Estrada ID: 923058731
Github: jjestrada2 CSC415 Operating Systems

# Assignment 4 – Word Blast

**Description**:
implement a concurrent word frequency counter in C using POSIX threads. The program takes a text file as input, divides it into chunks, and assigns each chunk to a separate thread. Each thread processes its assigned chunk, tokenizes the text, and updates a shared data structure to count the frequency of words. To ensure thread safety, a mutex is employed to protect critical sections. After all threads complete their tasks, the program sorts and identifies the top 10 most frequently occurring words with at least six characters. The execution time is measured, and the results are displayed, providing an efficient and parallelized solution for analyzing word frequencies in large text files.

**Approach / What I Did**:
Here is the to-do list of my approach:
Allocate and Initialize Storage Structures:

The program initializes and allocates memory for data structures like wordArray to store words and their frequencies.

Open File and Calculate Chunks:

The program opens the input text file specified in the command line arguments and calculates the size of the file. It then determines the size of chunks based on the number of threads.

Start Thread Processing:

Threads are created using POSIX threads. Each thread runs the wordCount function concurrently to process its assigned chunk of the file.

Wait for Threads to Finish:

The main program waits for all the threads to complete their tasks using the pthread_join function.

Process Top 10 Words:

After all threads finish, the program sorts the wordArray based on word frequencies to identify the top 10 most frequently occurring words.

Display Results and Timing:

The program prints the results, including the top 10 words and their frequencies, and provides the total execution time.

Cleanup:

The program closes the file, destroys the mutex (pthread_mutex_destroy), and frees allocated memory.

**Issues and Resolutions:**

I have the following issues:

1. Issue: Race conditions occur when multiple threads access shared data concurrently, and at least one of them modifies the data. The final outcome depends on the order of execution.

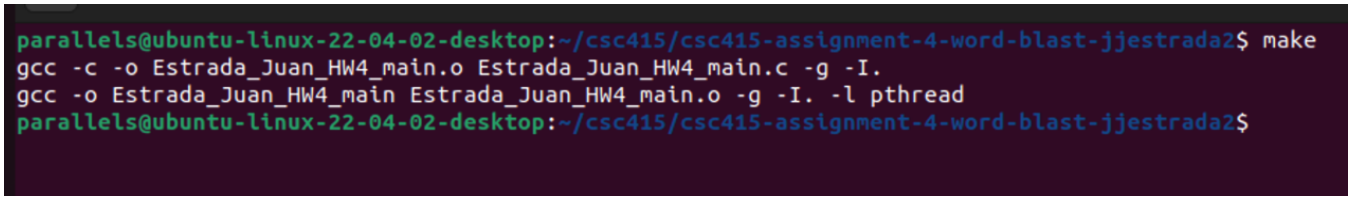
   Solution: Use synchronization mechanisms like mutexes to protect critical sections of code. This ensures that only one thread can access the critical section at a time.

2. Deadlock:

   Issue: Deadlocks happen when two or more threads are blocked forever, each waiting for the other to release a resource.

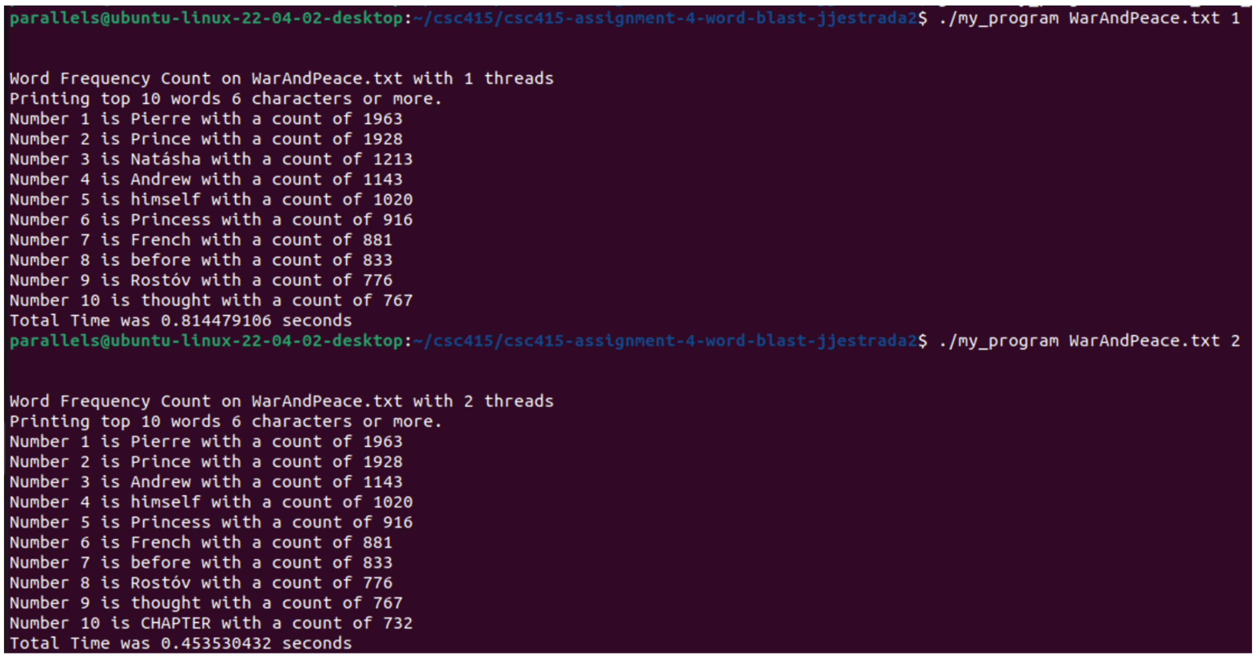
   Solution: Be cautious when acquiring multiple locks and ensure a consistent order. Use techniques like deadlock detection and prevention.

**Screen shot of compilation:**



```
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment-4-word-blast-jjestrada2$ make
gcc -c -o Estrada_Juan_HW4_main.o Estrada_Juan_HW4_main.c -g -I.
gcc -o Estrada_Juan_HW4_main Estrada_Juan_HW4_main.o -g -I. -l pthread
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment-4-word-blast-jjestrada2$
```

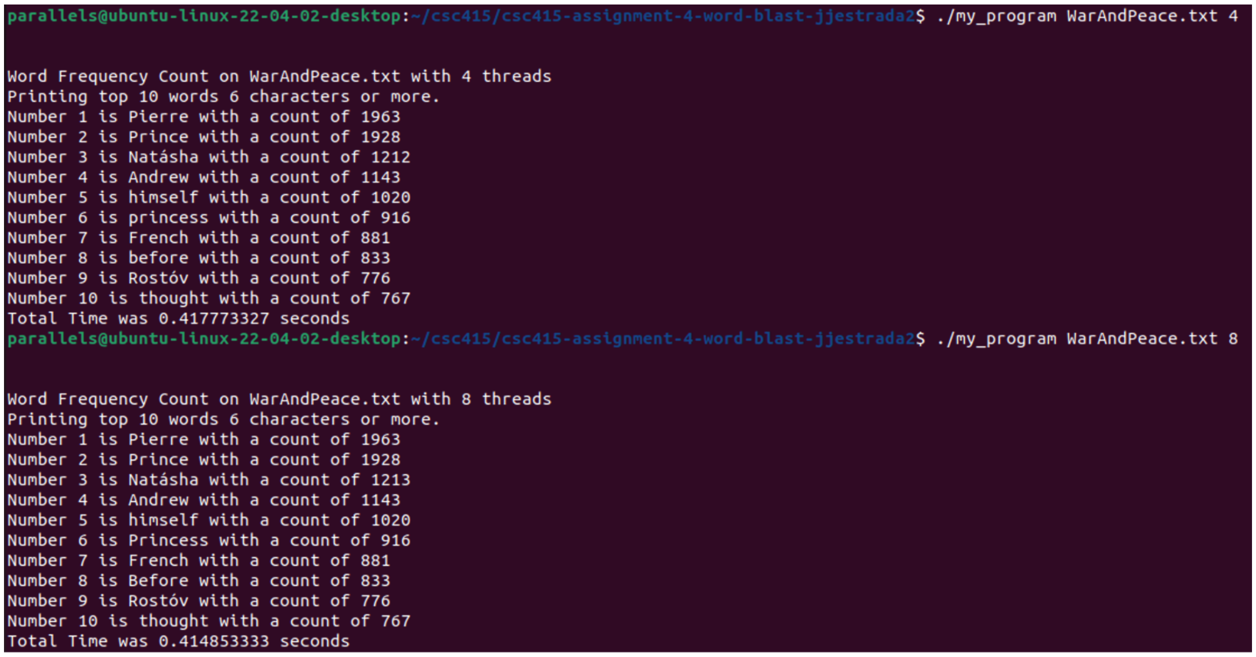
**Screen shot(s) of the execution of the program:**



```
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment-4-word-blast-jjestrada2$ ./my_program 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 0.814479106 seconds
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment-4-word-blast-jjestrada2$ ./my_program 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 Andrew with a count of 1143
Number 4 is himself with a count of 1020
Number 5 is Princess with a count of 916
Number 6 is French with a count of 881
Number 7 is before with a count of 833
Number 8 is Rostóv with a count of 776
Number 9 is thought with a count of 767
Number 10 is CHAPTER with a count of 732
Total Time was 0.453530432 seconds
```



```
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment-4-word-blast-jjestrada2$ ./my_program 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 0.417773327 seconds
parallels@ubuntu-linux-22-04-02-desktop:~/csc415/csc415-assignment-4-word-blast-jjestrada2$ ./my_program 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 0.414853333 seconds
```

Analysis:

1. Single Thread (1):

   Total Time: 0.814 seconds
   Top words are displayed based on their frequency.
2. Two Threads (2):

Total Time: 0.454 seconds
Some differences in the top words compared to the single-threaded run.
The total time has reduced, but the improvement is not significant.

3. Four Threads (4):

   Total Time: 0.418 seconds
   Similar top words as the two-threaded run.
   A further reduction in total time, but the improvement is diminishing.

4. Eight Threads (8):

   Total Time: 0.415 seconds
   Similar top words as the four-threaded run.
   The total time has slightly decreased, but the improvement is minimal.

Reflection:

Speedup and Scalability: As we increase the number of threads, the program's execution time generally decreases. However, the reduction in execution time diminishes as you increase the number of threads. This is a common behavior due to factors like overhead in thread creation, synchronization, and contention for shared resources (mutexes in this case).

Contention and Synchronization Overhead: As you increase the number of threads, contention for the mutex lock and the overhead of synchronization may start to impact performance. This is why the improvement in execution time becomes less pronounced with more threads.

File Reading Overhead: The program reads a file, and the file reading process might also contribute to the overall execution time. The performance gains from parallelization might be overshadowed by the file I/O overhead.