Sidney Thomas
Github: siid14
ID: 918656419
CSC415 Operating Systems

Assignment 4 – Word Blast

Description:

The provided code is a C program designed to read a text file, tokenize its content into words, and count the frequency of words containing six or more characters. This program use multithreading for processing.

The program begins by parsing command-line arguments to determine the input file name and the number of threads to use for processing.
It opens the specified file and calculates the file's size to divide it into segments for parallel processing.

Multiple threads are created, each responsible for processing a segment of the file. These threads tokenize the text, count word occurrences, and update a global data structure while ensuring thread safety through the use of a mutex.
After all threads have finished processing, the code sorts the word count data in descending order based on word frequency.

The top 10 words with their respective counts are displayed in a specified format, including word numbering.

Basic error handling is implemented to handle issues like file opening failures, memory allocation errors, and thread creation errors. Error messages are displayed to inform the user of any problems encountered.

The program measures the total execution time and prints it in seconds and nanoseconds using the clock_gettime function.

In summary, this C program provides a multi-threaded solution for analyzing word frequency in a text file, offering enhanced performance through parallel processing while maintaining proper error handling and timing measurements.

Approach / What I Did:

Here's a description of my approach:

In approaching the coding assignment for this word frequency counting program with multithreading, I followed a structured methodology to create a reliable solution. My approach can be summarized as follows:

Command-Line Argument Parsing: I began by parsing the command-line arguments to extract the input file name and the number of threads to be used for processing. This step ensured that the program could be configured to work with different input files and thread counts.

File Opening and Segmentation: After obtaining the input file name, I opened the file for reading using the open system call. To optimize processing and allow for parallelism, I determined the file's size using lseek and divided it into segments based on the number of threads. This segmentation ensured that each thread processed a distinct portion of the file.

Multithreading: For multiple threads, I created an array of thread data structures, each containing essential information such as the file descriptor, thread index, segment size, and a buffer for reading data. I used the pthread_create function to get multiple threads, and each thread executed the countWords function independently.

Word Tokenization and Counting: Inside the countWords function, I implemented a tokenization process using the strtok_r function, which split the text into words based on specified delimiters. I counted the occurrences of words with six or more characters. To ensure thread safety and prevent data races, I employed a mutex to protect shared data structures.

Error Handling: Robust error handling was integrated into the program. I checked for errors during file opening, memory allocation, thread creation, and file reading. When errors occurred, I used perror to display informative error messages and gracefully exited the program.

Sorting and Output: After all threads completed their tasks, I sorted the word count data in descending order based on word frequency using the qsort function. I then displayed the top 10 words and their counts in a user-friendly format as specified in the assignment.

Additionally, I closed the file descriptor and destroyed the mutex to prevent resource leaks.

User-Friendly Output: I designed the output to be informative and readable. The program displayed prompts, error messages, and the final word frequency counts in a clear and organized manner.

Issues and Resolutions:

During the coding assignment for this word frequency counting program with multithreading, I encountered a particular issue related to memory management, and I found a resolution to address it:

Issue: The issue I faced was related to memory allocation and deallocation for the thread-specific buffers. I allocated memory for each thread's buffer to read data from the file, but I needed to ensure that the memory was properly deallocated to prevent memory leaks.

Resolution: To address this issue, I implemented the following steps:

Memory Deallocation: After the threads finished processing and the word counting was complete, I added a loop to free the memory allocated for each thread's buffer. This ensured that no memory was leaked, and all resources were properly released.

Error Handling for Memory Allocation: I also implemented error handling for memory allocation when creating thread-specific buffers. If memory allocation failed for a thread's buffer, I displayed an error message and gracefully exited the program to prevent undefined behavior and crashes.

Also, occasionally, when I attempted to run the program immediately after a previous execution, it resulted in a segmentation fault. This unexpected behavior was unexpected because the program typically executed without errors.
I noticed that the issue appeared to be related to the timing of consecutive executions. When I waited for one or two seconds between program runs, the problem seldom occurred, and the program executed successfully with the expected output. However, when I attempted to execute the program immediately after the previous run, it occasionally resulted in a segmentation fault.

Additionally, for improvement to the initial version of the program, I incorporated the usage of strtok_r() for tokenization. This change was suggested by my teammates, and it

significantly improved the program's thread safety during tokenization processes. By utilizing strtok_r(), I ensured that the program could efficiently process text within multiple threads without any risk of data corruption or race conditions.

Analysis - Explain and reflect on why the times for the different runs are what they are, how does each run compare with the others

(Note: Screenshot of output + Execution time can found within the 'screenshot' part of this writeup)

1 Thread: Execution Time: 0.073650486 seconds

The single-threaded run is the fastest because it avoids the overhead of thread management and synchronization. However, it doesn't fully utilize all the hardware capabilities available to him.

2 Threads: Execution Time: 12.165220906 seconds

4 threads: Execution Time: 8.148069163 seconds

8 threads: Execution Time: 8.418819173 seconds

The multi-threaded runs (2, 4, and 8 threads) show slower returns as more threads are added. The overhead of creating and managing threads and dealing with shared data structures can offset the benefits of parallelism.
The choice of an optimal number of threads depends on factors like the file size, the complexity of processing, and the hardware architecture.

Screen shot of compilation:

Testing compilation

```
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-4/csc
415-assignment-4-word-blast-siid14$ make
gcc -c -o Thomas_Sidney_HW4_main.o Thomas_Sidney_HW4_main.c -g -I.
gcc -o Thomas_Sidney_HW4_main Thomas_Sidney_HW4_main.o -g -I. -l pthread
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-4/csc
415-assignment-4-word-blast-siid14$
```

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

- Testing with 1 – 2 – 4 and 8 threads

1 THREAD

```
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-4/csc
415-assignment-4-word-blast-siid14$ make run
./Thomas_Sidney_HW4_main WarAndPeace.txt 1



Word Frequency Count on WarAndPeace.txt with 1 threads
Number 1 is CHAPTER with a count of 740
Number 2 is Prince with a count of 102
Number 3 is Pávlovna with a count of 94
Number 4 is Pierre with a count of 69
Number 5 is princess with a count of 45
Number 6 is Andrew with a count of 44
Number 7 is vicomte with a count of 36
Number 8 is little with a count of 35
Number 9 is Vasíli with a count of 31
Number 10 is Hippolyte with a count of 29
Total Time was 0.073650486 seconds
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-4/csc
415-assignment-4-word-blast-siid14$
```

2 THREADS

```
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-4/csc
415-assignment-4-word-blast-siid14$ make run
./Thomas_Sidney_HW4_main WarAndPeace.txt 2


Word Frequency Count on WarAndPeace.txt with 2 threads
Number 1 is CHAPTER with a count of 735
Number 2 is Pávlovna with a count of 42
Number 3 is Prince with a count of 18
Number 4 is prince with a count of 16
Number 5 is himself with a count of 14
Number 6 is French with a count of 12
Number 7 is Balashëv with a count of 12
Number 8 is little with a count of 12
Number 9 is Vasili with a count of 11
Number 10 is always with a count of 10
Total Time was 0.023712016 seconds
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-4/csc
415-assignment-4-word-blast-siid14$
```

4 THREADS

```
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-4/csc
415-assignment-4-word-blast-siid14$ make run
./Thomas_Sidney_HW4_main WarAndPeace.txt 4


Word Frequency Count on WarAndPeace.txt with 4 threads
Number 1 is Pierre with a count of 1963
Number 2 is Prince with a count of 1583
Number 3 is Natásha with a count of 1225
Number 4 is Andrew with a count of 1145
Number 5 is CHAPTER with a count of 1096
Number 6 is himself with a count of 1018
Number 7 is French with a count of 884
Number 8 is before with a count of 780
Number 9 is Rostóv with a count of 776
Number 10 is thought with a count of 768
Total Time was 8.148069163 seconds
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-4/csc
415-assignment-4-word-blast-siid14$
```

8 THREADS

```
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-4/csc
415-assignment-4-word-blast-siid14$ make run
./Thomas_Sidney_HW4_main WarAndPeace.txt 8


Word Frequency Count on WarAndPeace.txt with 8 threads
Number 1 is Pierre with a count of 1965
Number 2 is Prince with a count of 1583
Number 3 is Natásha with a count of 1220
Number 4 is Andrew with a count of 1146
Number 5 is CHAPTER with a count of 1096
Number 6 is himself with a count of 1018
Number 7 is French with a count of 881
Number 8 is before with a count of 782
Number 9 is Rostóv with a count of 776
Number 10 is thought with a count of 766
Total Time was 8.418819173 seconds
student@student-VirtualBox:~/Desktop/Operating-Systems-CSC-415/Assignment-4/csc
415-assignment-4-word-blast-siid14$ make run
```