Threaded Word Frequency Counter In C

Joshua P. Costanza

Abstract

Threaded Word Frequency Counter in C

In this report, I will go over the functions and structure of the program. In class we were asked to create a program in C that would take the words from a text file and print out each words frequency in which it appeared in the document. We were also asked to implement pthreads during the analysis of the text document. I found that trying to implement this type of program in C was rather difficult for many reasons, but mostly due to the fact that there is no actual String data type within the language. Another problem with using C for this type of program is in order to obtain all the “words” from the text document; one must use a list. Since this is the case, I had to use a list type of structure in order to obtain all the characters from the text document and turn them into words in order to count their frequencies.

First I will discuss how the word counter logic itself works. At the top of the source code, “textReader,c”, you will notice the structure I have defined as “word”. This structure will be what is used to store the words and their frequencies, along with a pointer to the next word. These three data types are as follows: char \*str; int freq; struct word \*Next. The names are self-explanatory. Following the struct, I have written the prototype functions (interface) of the functions that will be needed to do the program. First of these four functions is “addWord()”. The function addWord will add a word to the list and will update the word as well. Second comes “printWords()”. This function will print the words and how many times it has occurred. Third is “getWords()”. The getWords function will be what grabs the words from the .txt file. Finally, the “Word\* createWordCounter()” function will create a new word counter structure. Following this last function prototype, you will see that I have declared a variable that will point to the first word counter in the list. This variable’s name is “point”. There is then two variables to count the total number of words and the amount of each individual word.

Following all the declarations is then the implementation of the prior prototype methods, addWord, printWords, getWords, and createWordCounter. Most of the work is done in createWordCounter. This method utilizes malloc() in order to compensate for the size of the words. Then the string from the pCounter will take the chars and use malloc() as well. Thus storing the words into word.

The addWord method works like this. First declare the variables for the pointers which are needed to keep order of the list. The names for these two variables are, as expected, “pCounter” and “pLast”. The method will then check if the pointer to the first counter within the list is null. If it is in fact null, it then sets itself to the call of the creareWordCounter method, passing the argument of word inside.

In method Main There is where most of the processing occurs. Firstly there is declared a pthread variable for thread 1. There is also a pointer for the file and a size variable which is set to the MAX\_TEXT\_LENGTH which was defined in the header above, also a status variable. The main method then searches for the text file inside the given location during the line file = fopen(“file.txt”, “r”). Then a buffer to hold the words in instantiated. While the pthread\_create is called, its arguments contain a call to the addWord method. This begins the process of adding the words from the specified file into the list. Therefore when the words are being added to the file it is being done in the stream of thread1. After all the words have been accounted for, there are three print statements that are made to show two things. First to show the total word count and the second to show the total number of individual words. Then all the words and their frequencies from the document are printed in the terminal. Followed by a closing of the file and a call to pthread\_exit.

I would also like to point out which files are contained within the .zip file for this program. The folder itself is titled “423\_Final\_Project”. Within the folder you will see four files. These files are as follows: 423\_final\_project\_report, myEssay.txt, textReader.c, textReader.o. There will also be a separate readMe file which will explain how to run the program itself. The readMe file will merely be an extension of this paper assessment.