TDQC

Word Sort

Jack Spence

28 July 2018

1. Write-Up

1.1 Requests

Requests from this program was to create a text parser that would sort word files into various ways. The ways to be sorted were alphabetical, by word score, by scrabble score, by length, as well as print only unique numbers. The words would be either supplied in a list of files or be typed directly into the command line with no arguments passed to the program. Another feature is a reverse option to reverse the output from ascending to descending order. A quirk that was requested was to have the reverse option be repeatable and only be enabled on odd number of requests and disabled for even requests.

1.2 Suggested Features

Suggested features attempted in my submission are to strip punctuation marks from the words as they are read in, design and use long options on the command line, as well as having a make clean and make debug option.

1.3 Syntax

Usage: ./ws [-h][-c NUM][-r][-n][-l][-a][-u][-s] [FILENAME]

-h, --help: This menu

-c NUM, --count NUM: Display NUM amount of lines.

-r, --reverse: Display results in reverse order.

-n, --number: Display results sorted by their number score.

-l, --length: Display results base on length.

-a, --alpha: Display results alphabetically(Default).

-u, --unique: Display results that are unique only.

-s, --scrabble: Display results based on their score in the game Scrabble.

-p, --punct: Strip punctuation from words in file.

If no parameters are set, the user can input data directly into the terminal.

To sort the input press CTL+D.

1.4 Sample Output

```
skydev@skydev:~/TDQC-Projects/wordsort$ ./ws head_english -n | head
counterrevolutionaries
electroencephalograph's
counterrevolutionary's
electroencephalographs
electroencephalogram's
electroencephalograms
electroencephalograph
counterrevolutionary
counterintelligence's
uncharacteristically
```

```
skydev@skydev:~/TDQC-Projects/wordsort$ ./ws head_english -s | head
quizzically
Chappaquiddick's
Chappaquiddick
Nebuchadnezzar
Czechoslovakian's
Czechoslovakians
Czechoslovakia's
Czechoslovakian
Schwarzkopf's
psychoanalyzing
skydev@skydev:~/TDQC-Projects/wordsort$
```

```
skydev@skydev:~/TDQC-Projects/wordsort$ ./ws head_english -s -r | head
c
cs
Tc
Sc
Ac
Ac
Ac
tic
sic
sac
skydev@skydev:~/TDQC-Projects/wordsort$
```

```
skydev@skydev:~/TDQC-Projects/wordsort$ ./ws head_english -l | head
electroencephalograph's
counterrevolutionaries
counterrevolutionary's
electroencephalogram's
electroencephalographs
counterintelligence's
electroencephalograms
electroencephalograph
chlorofluorocarbon's
counterrevolutionary
skydev@skydev:~/TDQC-Projects/wordsort$
```

2. Project Design Plans

2.1 Initial Design Plans

I only faintly knew what I was going to start with, which was storing the input in a binary search tree. I thought it would be pretty easy to implement as well as it will sort the data as necessary as the tree is built. I had never implemented one before so I knew this would be a good challenge for me.

2.2 What didn’t work

One of my biggest hurdles was to make everything dynamic, and allow the user to use as big of a file they wanted as well as have each line be of whatever length it happened to be. I took me quite some time to get the logic down to have a malloc’d char array in my node struct. As well when and where to free that memory.

2.3 What went well

Getting the binary search tree happened rather quickly once I had the initial node design down. I think it went pretty smoothly for my first time implementing one. The logic I designed to propagate the tree could use some improvement, but works in the end.

2.4 Conclusion

The wordsort project was quite the learning experience for me. It has been the most time consuming project as well as the most technically demanding project I have ever completed. I look forward to taking the lessons learned from this project forward into my development.