KOÇ UNIVERSITY

COMP 201 - Spring 2022
Assignment 2 - Strings in C: Spell Checker

Assigned: 11 March 2022 23:59, Due: 24 March 2022 23:59

Nafiseh Tofighi (ntofighi21@ku.edu.tr) is the lead person for this assignment.

1. Introduction:

The purpose of this assignment is to advance your skills in manipulating strings and learn to use string functions efficiently on C. You will do this by working on a real-life problem: A spell checker. You will use Natural Language Processing methods for this assignment. You are provided with a dictionary of correct English words to use as a source. For this assignment, you will implement a program that prepares sentences to process, detect misspelled words, and suggest correct words instead.

Good luck, and welcome to the Spell checker assignment!

2. Logistics

This is an individual project. All handins are electronic. Clarifications and corrections will be announced on Blackboard.

3. Handout Instructions:

Accept the GitHub Classroom assignment using the link: https://classroom.github.com/a/MczMv599. Clone the GitHub repository created for you to a Linux machine in which you plan to do your work (We advice you to do your work on our linux servers [linuxpool.ku.edu.tr]. See Section 8 for details.) :

```
git clone https://github.com/COMP201-Spring22/assignment-2-USER.git
```

(Replace USER with your GitHub username that you use to accept the assignment) For this assignment, you need only modify the main.c file, and at the end, submit main.c file.

- **main.c:** The main file of the assignment that should be modified and submitted to be graded.
- **Dictionary:** List of correct English words.
- **sample.txt:** Contains sample sentences to be used as input of the program.

4. Tasks:

This assignment contains seven main parts and a bonus part. After completing each part, you can compile your code to find out if everything works well or not. With the help of the below command, you may compile the code:

```
gcc -o OUTPUTFILE main.c
```

Then pass the `sample.txt` file as an input argument to run the output file.

```
./OUTPUTFILE sample.txt
```

4.1 To Lower:

The first part is related to the convert strings into the lower case without enjoying the standard string functions. It means you are not allowed to use strlwr() in this part. This is important that all letters be in lowercase; this helps the program not make a difference between two exact words in different cases. For example, if this function is not applied, 'Apple' and 'apple' may be considered as two separate words. This part should be implemented in the `toLower()` function and the input argument is `input` string. You should return the `input` which contain the same text that all letters are in lowercase format. For instance:

INPUT: This is a 'TEST' sentence!
OUTPUT: this is a 'test' sentence!

4.2 Remove Punctuations:

In order to detect wrong words, the text must include only letters. As a result, you need to implement this function to remove all punctuations for the second step. Punctuation such as '., :, !, ?, etc.' may cause misunderstanding for the spell checker, so the output of this function should be only letters. Likewise, previous section, this function's input argument is string `input`, and it should return exact text without any punctuations.

INPUT: this is a 'test' sentence!
OUTPUT: this is a test sentence

4.3 Tokenization:

The next phase is Tokenization, which splits the input string into words to process them. Under this circumstance, we can check each word if it is correctly spelled or not. For this part, sentences are available at string `input` and these sentences should be split. The array named `tokens` is defined. Try to store each word as an element of this array. You are NOT allowed to use the strtok() function in this part. After implementing this part, you should be able to see your tokens at the output.

INPUT: this is a test sentence
OUTPUT: 'this', 'is', 'a', 'test', 'sentence'

4.4 Detecting Misspelled Words:

There is a file named `dictionary` available in the assignment folder, which is a list of correct English words. For step number four, you should search for each word to determine whether it is in the dictionary list or not. In other words, if a word is in this list means it is correctly spelled, and if not, it means it is a misspelled word. In this part status of all words will be determined that they are correct or misspelled.
previously, the `dictionary` file is loaded in `dictionary` array, in which every word is stored as an element of this array. So, you can simply use it. According to the previous part, the words are available in `tokens` array. As the output of this part, you should put wrong words as elements of `incorrects` array. After implementing this part, you should be able to see incorrect tokens at the output.
Hint: variable `used` contains number of pointers used for storing `dictionary` words so a possible loop instruction could be: `for (size_t j = 0; j < used; j++)` In this case `dictionary` elements

available at `dictionary[j]`.

INPUT: 'this', 'is', 'a', 'testt', 'sentence', 'with', 'rong', 'testt', 'words'
OUTPUT: 'testt', 'rong', 'testt'

4.5 Most Common Misspelled Word:

Now we have a list of wrong words in `incorrects` array. In this step, we want to find the word that is mostly spelled wrong and store it in maxElement array. In cases that the max repeated word is more than one word, report all commonly misspelled words as elements of maxElement array. After implementing this part, you should be able to see the most repeated incorrect words at the output.

INPUT: 'testt', 'rong', 'testt'
OUTPUT: 'testt'

4.6 Misspelled Words Distributions:

For This part, calculate how many percent of the errors belong to n-letter words. For example, how many wrong words are 2-letter, how many are 3-letters, etc.? For $n < 10$, keep the percent of the n-letter words in the n-th element of the `percent(type:int)` array. For $n >= 10$, sum all their percentage and report them as the 10-th element of the `percent` array. After implementing this part, you should be able to see the percent of each category at the output.

INPUT: 'testt', 'rong', 'testt, 'categori'
OUTPUT: '4-letter: 25%', '5-letter: 50%', '8-letter: 25%'

4.7 Predicting Correct Words:

At the last main part, we want to suggest relevant, correct words to replace with misspelled words. There are different ways to implement this, but suppose that the wrong letter is the last letter of the words. For example, if the wrong word is 'appla,' implement this part of the code to look for all correct words in the `dictionary` list, which starts with 'appl.' In this case, the 'apple' is one of the correct suggestions. Store each suggestion as an element of `suggestions` array. After implementing this part, you should be able to see suggested words at the output.

INPUT: 'anglef'
OUTPUT: 'angle', 'angled', 'angler', 'anglers', 'angles', 'anglesey', 'anglesite', 'angleworm'

4.8 BONUS: Levenshtein Distance

In order to find the best suggestions, the distance between the incorrect word and all correct words should be calculated. Then the correct words that have minimum distance with incorrect words could be the best suggestion. One of the common metrics used to calculate the distance between two words is 'Levenshtein Distance.' For the bonus part, try to implement details of this metric. Store Levenshtein suggestions for `incorrects` list as elements of the `levenSuggetions` array.

5. Submission

As with Assignment 1, we use GitHub for the submissions as follows. Note that we want you to get used to using a version management system (Git) in terms of writing good commit messages and frequently committing your work so that you can get most out of Git.

1. Commit all the changes you make:

```
git commit -a -m "commit message"
```

   Note: please use meaningful commit messages because

2. Push your work to GitHub servers:

```
git push origin main
```

You are expected to submit a `main.c` file. Please accurately read and follow the instruction mentioned in the `main.c` file as a comment. Enter your codes only between the specific spaces prepared at the `main.c` file. And please do not modify commented lines in the source code file.

6. Evaluation:

Your score will be computed out of a maximum of 100 points based on the following distribution:

- To Lower: 10%
- Remove punctuations: 10%
- Tokenization: 20%
- Detecting Misspelled Words: 20%
- Most Common Misspelled Word: 10%
- Misspelled words distributions: 10%
- Predicting Correct Words: 20%
- BONUS: Levenshtein Distance: +10%

Effective use of version control: You are required to push your changes to the repository frequently. If you only push the final version, even if it is implemented 100% correctly, you will lose a fraction of the grade because you are expected to learn to use Version Control Systems effectively. You do not have to push every small piece of change to Github but every meaningful change should be pushed. For example, each of the tasks coded and tested can be one commit. For each task, there should be at least one commit (with proper commit message) that includes just modifications on that function.
Important Note: We use automated plagiarism detection to compare your assignment submission with others and also the code repositories on Github and similar sites. Moreover, we plan to ask randomly.

7. Oral Assessment

Important Note: We plan to ask randomly selected 10% of students to explain their approach verbally after the assignments are graded. And one may lose full credit if he or she fails from this oral part.

8. How to use linuxpool.ku.edu.tr linux servers

I Connect to KU VPN (If you are connected to the KU network, you can skip this step.)
See for details: https://confluence.ku.edu.tr/kuhelp/ithelp/it-services/network-and-wireless/vpn-access

II Connect to linuxpool.ku.edu.tr server using SSH (Replace USER with your Koç University username):

```
ssh USER@linuxpool.ku.edu.tr
```

(It will ask your password, type your Koç University password.)

III When you are finished with your work, you can disconnect by typing: `$ exit`

Your connection to the server may drop sometimes. In that case, you need to reconnect.

We advise you to watch the following video about the usage of SSH, which is used to connect remote servers, and SCP, which is used to transfer files between remote servers and your local machine: https://www.youtube.com/watch?v=rm6pewTcSro

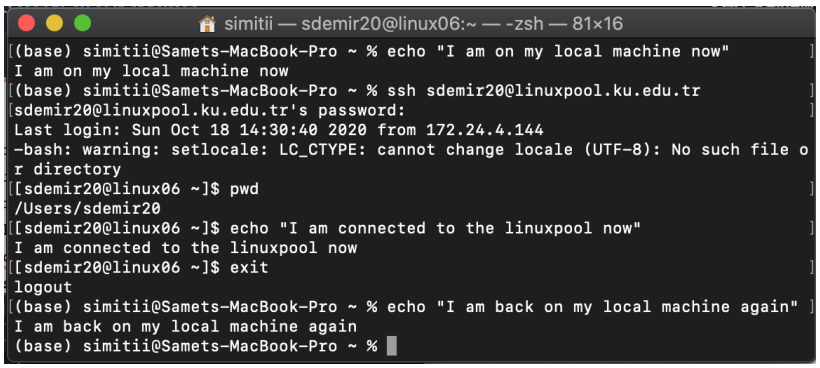

Figure 1: How to connect and disconnect using SSH

9. Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss the given assignments with your classmates, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the web material as everything on the web has been written by someone else. See Koç University - Student Code of Conduct.

10. Late Submission Policy

You may use up to 7 grace days (in total) over the course of the semester for the assignments. That is, you can submit your solutions without any penalty if you have free grace days left. Any additional unapproved late submission will be punished (1 day late: 20% off, 2 days late: 40% off) and **no submission after 2 days will be accepted.**