KOÇ UNIVERSITY

COMP 201 - Fall 2023
Assignment 1 - Strings in C

Assigned: 18 October 2023 23:59, Due: 1 November 2023 23:59

Doğa Kukul (dkukul19@ku.edu.tr) is the lead person for this assignment.

1 Introduction

This assignment will enhance your understanding of dynamic memory management, file I/O operations, and string manipulation in C programming, which are of paramount importance for handling real-world data processing tasks. You are going to analyze the frequency of words that occur in spam messages and create a simple spam filter for SMS messages.

You are provided a dataset, The SMS Spam Collection, where each line consists of a label indicating whether the message on the same line is a spam message or not. You will process the preprocessed version (`preprocessed_dataset.txt`) file and do a frequency analysis.

2 Datasets

`SMSSpamCollection.txt` This dataset contains labelled messages. It is only for you to observe and get an idea of what kind of information is stored in the data. It has classes and messages on each line. A line begins with either 'spam', indicating that the following message is a spam message, or 'ham', indicating that it is not a spam message. The labels are followed by a tab (\t) character that separates the classes and messages. After the tab character, the message starts. For simplicity, you may assume that each word in the message is seperated with a space character (" "). At the end of each line, there is a new line (\n) character.

`preprocessed_dataset.txt` This is the preprocessed version of the SMSSPamCollection.txt. Mainly, words that contribute little information (called stopwords) and the punctuations are removed from the original data.

`testData.txt` is a small excerpt from the preprocessed SMS Spam Collection dataset. You will use this file to see whether your spam detector works well by comparing the output of the detector to the ground truth values in the file.

3 Logistics

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

4 Handout Instructions

4.1 How to start

I Accept the GitHub Classroom assignment using the link: https://classroom.github.com/a/66N_6tjo

II 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/COMP-201-Fall-2023/assignment-1-USER.git`
(Replace USER with your GitHub username that you use to accept the assignment)

III **[IMPORTANT]** After cloning the repository, you are required to write the following honor code in a new file called "honor.txt" and commit & push it: "I hereby declare that I have completed this assignment individually, without support from anyone else." You can use the following command to create "honor.txt" file and write the honor code in it:
`$ echo "I hereby declare that I have completed this assignment individually, without support from anyone else." > honor.txt`

4.2 Main Task

In the environment that you set up, create a main.c file. You will do all of your coding in this file. To run your code, simply run the "make" command in the directory you're working on.

4.3 How to Read a File

- First, open the text file using the `fopen()` function.
- Then, use `fgets()` function to read text from the stream, and store it as a string. The newline of EOF character stops `fgets()` so you can check the newline or EOF file character to read the whole line. **Note:** `fgets()` also reads endline characters, and you should ignore newline character at the end of the sentence.
- When you are done with the file, close the file with the `fclose()` function.
- For more information, you can refer to these links: Link 1, Link 2

5 Tasks

Task 1: Count Occurrence

In this task, you will prompt the user to enter a word, and display how many times the word appeared in spam and non-spam messages. The search should be case insensitive.

(Hint: a word may appear more than once in a message)

Sample Outputs

```
$ make
Please enter a word to search for:  computer
The word 'computer' appears 6 times in ham messages and 5 times in spam messages.
$ make
Please enter a word to search for:  comp
The word 'computer' appears 0 times in ham messages and 5 times in spam messages.
$ make
Please enter a word to search for:  price
The word 'price' appears 7 times in ham messages and 18 times in spam messages.
$ make
```

```
Please enter a word to search for:  terminal
This word doesn't occur in the text!
```

Task 2: Identifying Most Common Spam Words

In this task, you will identify the 10 most occurring words in the spam messages. To do this, you may need to define a new struct to keep a count of unique words and their occurrences. Once you identify the most common 10 words, print them in descending order.

Sample Outputs

```
$ make
.
.
.
Top 10 occurring words in spam messages:
call:  292 times
free:  204 times
txt:  149 times
u:  148 times
ur:  142 times
.
.
.
```

Task 3: Writing a Spam Detector

Your final task is to use the most common 10 words that you identified in Task 2 to label a message as spam or ham. You should open the `testData.txt` file and read the lines just like in the previous parts, but this time, implement a function that checks whether the message contains at least 3 of the words that are in the "spam list". If that is the case, the function should return "spam", otherwise it should return "ham".

Finally, compare the outputs of your function to the actual labels in the data and report the accuracy of your detector.

$$\textbf{Accuracy} = \frac{\text{\# Correct Identifications}}{\text{\# Total Identificaitons}} \tag{1}$$

(Note that the total identifications here refers to the total within their respective groups (e.g., total spam identifications or total ham identificaitons))

Sample Outputs

```
$ make
.
.
.
Accuracy of spam predictions:  ##.##%
Accuracy of ham predictions:  ##.##%
Total spam messages:  14
```

```
Total ham messages:  83
.
.
.
```

Task 4: Observations

Did the spam detector perform as well as you expected? If not, what could be the reason for this behavior? What would you change to improve the detector? Write your answers as a comment at the bottom of your code.

6 Evaluation

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

15 Task 1.

35 Task 2.

35 Task 3.

5 Task 4.

5 Style points.

5 Effective use of version control points.

Effective Use of Version Control Points. 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 functions 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 task.**

Style Points. Finally, we've reserved 5 points for a subjective evaluation of the style of your solutions and your commenting. Your solutions should be as clean and straightforward as possible. Your comments should be informative, but they need not be extensive.

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 selected 10% of students to explain their code verbally after the assignments are graded. And one may lose full credit if he or she fails from this oral part.

7 Handin Instructions

As with Assignment 0, 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.

I Commit all the changes you make: `$ git commit -a -m "commit message"`
Note: Use meaningful commit messages as in huge projects they become really helpful. Try to gain this habit from early on.

II Push your work to GitHub servers: `$ git push origin main`

8 How to use linuxpool.ku.edu.tr linux servers [1]

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 advice 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

9 Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, 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 material found on the web 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.**

[1] For details, please see the guide on linuxpool that we have announced on Blackboard