Lab 6: Words, Words (and scripts and structs)

Given a text, what is its vocabulary?

That is, how many words are used. For example, the entire text of the children's book, *Green Eggs and Ham* by Dr. Seuss, contains only 50 words.

In this lab we will try and answer two questions about texts:

1. How many words are used in a given text?
2. What are the N most frequent words in a given text?

To answer these questions, you will write a C program and also, solve problems by some of the Linux commands and utilities.

1. How many words in a given text?

Whenever one has to process text, the tricky question is dealing with punctuations. For example, in the sentence below:

O'er the ramparts we watched, were so gallantly, streaming?

The apostrophe in the word, O'er, could be replaced by a 'v'. But what about words like it's, didn't, etc.? Properly addressing these comprehensively requires deep knowledge of syntax rules which can then be incorporated in a program. Additionally, if we're only interested in counting words, we have to normalize words spelled in upper/lowercase letters. For example, a "The" at the start of a sentence is the same word as "the" elsewhere in a sentence.

In order to simplify issues surrounding punctuations and capitalizations, we will make the following decision: before processing any text, we will **replace all punctuations with a blank character** (thus "O'er" will become "O er"), and **we will convert all uppercase letters into lowercase**. This will obviously result in an approximation, but will nevertheless get us started without too much loss, as you will soon see.

Linux has some very useful commands that come in handy when solving problems. With a good knowledge of the command set, you do not always have to write programs to address all processing tasks. The commands enable us to process data much more efficiently and correctly.

`tr – translate or delete characters`

This utility "translates" input it gets from standard input. For example:

```
cat xxx | tr A-Z a-z
```

converts all uppercase letters in the input, into lowercase letters. Go ahead and enter the command in your shell and enter some text to see how it works.

Let's take the *Green Eggs and Ham* text (in `~gtowell/Public246/Lab6/ham.txt`) as our test input. First, examine its contents (using `cat/more/less`). Next, try the command:

```
cat ~gtowell/Public246/Lab05/ham.txt | tr A-Z a-z
```

You will see that all the text is now converted into lowercase letters. Next try the command:

```
cat ~gtowell/Public246/Lab05/ham.txt | tr [:punct:] " "
```

All punctuation disappears and is replaced by a space character (“ “). Put the two tr command together to eliminate all punctuation and convert to lower case.

For the next task, in order to count the number of words, we can again, instead of writing a program, use `tr` yet again. For example, the first two lines of ham.txt are (what is the unix command for just printing the first two lines?):

```
I am Sam
I am Sam
```

This should be printed as:

```
I
am
Sam
I
am
Sam
```

What is the `tr` command for splitting words into lines in this way?

Now, put these three tr commands together to get the text, downcased, without punctuation; one word per line.

Finally, to get the number of words in the document, pipe the output through `sort` then through `uniq`

Lastly, all you have to do is count how many words are in the output. How?

For ham.txt you should get 50 (or 51)..

You now have a chain of commands to accomplish the task. Linux allows you to define a command of your own that will carry out all the commands in the chain as needed. This is called a script. To create a simple script, create a new file named, getwords, with your piped commands. It should look like (this is incomplete):

```
#!/bin/bash
tr [:punct:] ' ' | tr A-Z a-z
```

Make sure the above file is executable:

```
chmod 700 getwords
```

You can now run the entire set of commands by entering just the name of the script file:

cat ham.txt | ./getWords | wc

This is your first Linux script! We will experiment more with scripts later in the course.

Legend has it that someone challenged Dr. Seuss that he couldn’t write a story using just 50 words. His book, *Cat In The Hat*, uses only 236 words. Compare this to the vocabulary used in The Pickwick Papers (Dickens) or the Descent of Man (Darwin). Both are in ~gtowell/Public246/Lab06

Copy your script to:

`/home/gtowell/submissions/fall2019/cmsc246/lab6/YOURLOGIN_script`

2. What are the N most frequent words used in a given text?

First modify your getwords script to remove the `wc` and `uniq` (keep the sort if it was there)

Now write a program that processes the output from the bash script to get the N most common words . Here is the start of an algorithm to get the N most frequent words:

```
While there are words (in standard input):
    count the number of times the current word appears
    find the word with the smallest count
    if current word count > smallest count
        replace smallest with current

Print the N words
```

You can do this all just reading from standard input using the script from part 1.

This task is an excellent opportunity to define and use your first struct. Here is a struct defintion you might find useful:

```
typedef struct {
    char word[50];
    int count;
} Words;

Words words[N];

for (int i=0; i<N; i++)
     words[i].count=0;
```