Document Version: 1.0 (a newer version number means an update on the project)
Assignment Date: 1/12/2017
Due Date - No Penalty: 2/2/2017, 11:59pm
Due Date - 10 Points Penalty: 2/9/2017, 11:59pm

Objectives:

- Warm up with C programming
- Practice with data structures, dynamic memory allocation, and pointers
- Practice with systems calls (i.e. related library functions).
- Exercise reading the man pages in Unix
- Trace the systems calls that a program executes.
- Discipline your programs for black-box testing where the expected output will have quite rigit formats.

1 Project Description

Write a C program in Linux that will read two input files, find the common words passing in both files if their number of occurrences are higher than or equal to the specified frequency limit for both files, sort the common words using the insertion sort algorithm, and print the sorted words into an output file together with their total number of occurrences for both files. For each input file, your program will build a separate *linked list* where each node of the list will contain a unique word passing in that file and the number of occurrence of that word (hereafter referred to as *count*) within that file. Hence, each node of the linked list will be a **`struct`** having fields to store the word itself (`char *`) and its count (`int`). You will build the list as a ***doubly linked list***, where each node in the list has a pointer to the next node and the previous node.

After building the lists, your program will find the common words passing in both of the lists if their *count* is larger than or equal to the specified frequency limit for both files, find their total count considering both files, and print this information into an output file in ascending sorted order. Sorting will be done based on the *word* field. For comparison of the words, you will use the `strcmp()` function. **It is the requirement of this assignment that you will implement *insertion sort* algorithm for sorting.** Each line of the output file will contain information about a common word passing in both input files in the following format:

Word,TotalCount

Note the comma (instead of space or TAB characters) between the fields. The output should not contain any spaces, TABs, or empty lines. So, after you write the last line of information, you should close the file, and this should be the end of your program. **It is very important that you produce the output in this rigid format since we will use this format in our automated black-box tests.** Words in input files are separated by whitespace characters, which can either be a <SPACE>, a <TAB>, or a <NEWLINE> character.

2 Development

It is the requirement of this assignment that you will use doubly-linked lists and dynamic memory allocation. In this way, you will be able to practice with malloc, pointers, and structs. The name of your executable file has to be **`ffcw`** indicating find frequent common words. A sample invocation of your program can be like the following:

```
./ffcw in1.txt in2.txt limit out.txt
```

Here, `in1.txt` is name of the the first input text file, `in2.txt` is the name of the second input text file, `limit` is the frequency limit to be used, and `out.txt` is the name of the output text file which you will store your final output. Note that input and output file names do not have to be `in1.txt`, `in2.txt`, and `out.txt`, different names can be passed to the program as command line arguments. **It is very important that you follow the specifications so that you do not lose any points.**

Example 1 *An example first input file (for example, `in1.txt`) can be like the following:*

```
THE GODFATHER
_____________

Screenplay

by

MARIO PUZO

and

FRANCIS FORD COPPOLA










THE THIRD DRAFT                           THE PARAMOUNT PICTURES
                                          1 Gulf and Western Plaza
March 29, 1971                            New York, New York 10019
```

and an example second input file (for example, `in2.txt`) can be like the following:

THE GODFATHER

Part Two

Screenplay by

Mario Puzo

and

Francis Ford Coppola

THE SECOND DRAFT

September 24, 1973

PARAMOUNT PICTURES
1 Gulf and Western Plaza
New York, New York 10019

Assuming the following invocation:

```
./ffcw in1.txt in2.txt 2 out.txt
```

Then the output file `out.txt` *will be as follows:*

```
NEW,4
THE,5
and,4
```

You will develop your program in a Unix environment using the C programming language. You can develop your program using a text editor (emacs, vi, gedit etc.) or an Integrated Development Environment available in Linux. *gcc* will be used as the compiler. You will be provided a Makefile and your program should compile without any errors/warnings using this Makefile. Black-box testing will be applied. Your program's output will be compared to the correct output. A simple black-box testing script will be provided to you through Blackboard for your own testing; make sure that your program produces the success message in the provided test. However, we may not use the provided test for your grading. More complicated tests (possibly more then one test) may be applied to grade your program. Submissions not following the specified rules here will be penalized.

3 Tracing the System Calls

After finishing your program, you will trace the execution of your code to find out all the system calls that your program makes. To do this, you will use the `strace` command available in Linux. See `man strace` for more details on `strace`. In a separate README file, you will write **only the names** of the system calls that your program made in ascending sorted order by eliminating duplicates if any.

4 Checking the Memory Leaks

You will need to make dynamic memory allocation. If you do not deallocate the memory that you allocated previously using **`free()`**, it means that your program has memory leaks. To receive full credit, your program should be memory-leak free. You can use `valgrind` to check the memory-leaks in your program. `valgrind` will output:

"All heap blocks were freed - no leaks are possible"

if your program is memory-leak free.

5 Submission

Submission will be done through Blackboard strictly following the instructions below. Your work will be penalized 5 points out of 100 if the submission instructions are not followed. Memory leaks and compilation warnings will also be penalized with 5 points each, if any. You can check the compilation warnings using the **-Wall** flag of gcc.

5.1 What to Submit

1. `README.txt`: It should include your name, ID, and the list of system calls that your program made. **Only unique the names of the system calls should be written in ascending sorted order where each line has a single system call name, no other information should be provided.**
2. `ffcw.c`: Source code of your program.
3. `Makefile`: Makefile used to compile your program.

5.2 How to Submit

1. Create a directory and name it as your UofL ID number. For example, if a student's ID is 1234567, then the name of the directory will be 1234567.
2. Put all the files to be submitted (only the ones asked in the *What to Submit* section above) into this directory.
3. Zip the directory. As a result, you will have the file 1234567.zip.
4. Upload the file 1234567.zip to Blackboard using the "Attach File" option. You do not need to write anything to the "Submission" and "Comments" sections. NO LATE SUBMISSIONS WILL BE GRADED!
5. You are allowed to make multiple attempts of submission but only the latest attempt submitted before the deadline will be graded.

6 Grading

Grading of your program will be done by an automated process. Your programs will also be passed through a copy-checker program which will examine the source codes if they are original or copied. We will also examine your source file(s) manually.

7 Changes

- No changes.