Assignment 2 - Colsort

Due: 09/20/2023

This assignment will require you to build on what you've learned previously to build an new application called `colsort`.

Instructions

Write a C program named `colsort.c` that accepts two command line arguments: the first command line argument, is the index of the nth word, and the second argument is file to be sorted.

The program should be invoked as follows:

```
$ ./colsort -<nth_word> <input_file_name>
```

Where $-$`nth_word` is an integer, and is always prefixed by a dash. Passing an index of one, the first word should be selected in the line. If no nth word supplied, then it should default to one. The second argument, the file name, and is always required. If it is not supplied, the program should exit with an error message specifying why. Additionally, each line will be limited to **128 characters**, but will often be be less then that. If the specified nth_word does not exist in a given line, your program should instead use the last available word. Lastly, make sure your string comparisons are case-sensitive.

Next, your program should then perform the following tasks

1. Read the lines from the input file.
2. Sort the array alphabetically (case-sensitive) based on the nth word.
3. Print the sorted lines to the console.

Examples

Suppose you have two input files named `file1.txt` with the following content:

```
this line is first
but this line is second
finally there is this line
```

If you run your colsort and give it this file as input, it should print:

```
but this line is second
finally there is this line
this line is first
```

because "but" is alphabetically before "finally" is before "this". If, however, you pass in a flag to sort a different key, you'll get a different output. For example, if you call colsort 2 on this file, you should get:

```
this line is first
finally there is this line
but this line is second
```

because "line" comes before "there" comes before "this". Yes, we are assuming -2 means the second word in each line (like most people would, except computer scientists who always want to start at 0).
As part of your testing process, aim to use a wide variety of test cases. You can generate test cases programmatically or find sample files from the internet. Test with files of different sizes, empty files, files with special characters, and files in different directories. The goal is to ensure that your program handles various scenarios accurately and reliably.

Linux File Operations

Here are some recommended C functions that you should use in your `colsort` program to achieve the desired functionality. Along with the functions from the last assignment, you'll want to look into using these functions.

It's worth mentioning that these are some functions that can be used to complete this assignment. The functions listed are not required, just a suggestion. You may find out that implementing your own version which better suits your code might be the best. If you think of a different way of completing the assignment, by all means do so.

- `char* strtok(char *str, const char *delimiters)`: Tokenizes a string into substrings based on specified delimiters. Useful for finding the nth word
- `void *malloc(size_t size)`: Allocates memory on the heap, used to create an array.
- `void free(void *ptr)`: Deallocates memory previously allocated by `malloc`.
- `size_t strlen(const char *str)`: Calculates the length of a string.
- `int strcmp(const char *str1, const char *str2)`: Compares two nth words
- `void qsort(void *base, size_t nmemb, size_t size, int (*compar)(const void *, const void *))`: Sorts an array using a specified comparison function.

If you encounter any issues while using the C functions mentioned in this assignment, you can refer to the man pages for detailed documentation on these functions. Additionally, you can find more information in chapter 2 of the textbook. When using these functions, make sure to include the appropriate header files (`stdio.h`, `stdlib.h`, `string.h`) and check for errors after each operation. It's important to handle errors gracefully and provide informative error messages to the user.

Submission

Submit the following files in a compressed archive (e.g., `username-assignment2.tar` or `username-assignment2.zip`) via canvas

- `colsort.c`: Your C program implementation.
- `Makefile`: A Makefile to compile your program. It should produce an executable named `colsort`.
- `README`: A short summary of your implementation, explaining how your program works.