Group:

David Ha : dth69

Michael Martinez : mbm193

NOTE: In order to compile the program, use the command “make ww” using the provided MakeFile

For this program to be correct, the program must be able to successfully wrap text coming from either standard input, a text file, or a directory containing text files. In the first case and second case, the program must successfully take the second argument (number of columns) and wrap the text to this limit and print to standard output. For the third case, the program must successfully wrap the text of files contained in a given directory to the given number of columns and write the results into new files. One simple way in which we tested if the program would provide the desired output is to compare the text of the desired results to the actual results outputted by the program. If the results were to match, then the program would be considered successful. Also, to ensure that the produced text is normalized, the text results of a wrapped file and a re-wrapped file were compared to see if there were any differences between the two results. If the command were to print nothing, this would mean that the text is successfully normalized.

In order to ensure that the program works correctly, we tested for the following cases and applied these solutions if applicable:

a. Testing of program after compiling with fsanitize to ensure no heap overflow or memory leaks.
    - For this test we just had to ensure that we malloc’d the correct sizes and for every call to malloc/calloc we free’d the associated data type.
b. Given number of columns is a negative number
    - For this test we included checks performing “atoi(argv[1]) > 0” assuming that the first given argument after the file name was the wrap width.
c. Given text file or directory does not exist
    - After checking the directory for any entry with the same name as provided by argv[2], our programs waits to read stdin for text to wrap instead.
d. The number of arguments is not sufficient
    - There are two cases. The first is that the file must have at least 2 arguments so that we can at least establish the necessary width from argv[1]. The other is that there can only be 3 total arguments therefore our code checks for 1 < argc < 4.
e. Text file contains words that exceed the given max length
    - In wrap_file we keep track of the total length of the word and cross check it with the given width. If the written word's length is longer than the max length of a wrapped line we modify the status to EXIT_FAILURE so that when we return the

status it will not be a success. In debugging we report the word that exceeded max length.

f. Consecutive spaces within a line of text
    - For this check we always check for the letter in the buffer using isspace() from ctype header to look for new lines and spaces. If it returns false we keep adding to our word, however if it returns true then we know that the word is complete and we add it to our output file. We never add a space to a word, and so when we check for the written value of the current word, if it is 0 and we are checking a space from the buffer we just continue.
g. The text file is empty
    - Since our program should accept any file based on instructions in the Output section, it will still take an empty text file and run it through wrap_file but only return a new line in the stdout or 'wrapped' text file.
h. Considering cases specific to directories:
    - Any existing wrapped files are overwritten in the case that an already wrapped file is wrapped once more
        1. In the case that a text file has been already wrapped previously by the program, the program would proceed to overwrite the results of the previous iteration of the program to prevent any duplicate wrapped files
    - Text files that start with "." or "wrap."
        1. The program simply checks the name of the text file to ensure that the name of the file does not start with the substrings "." or "wrap".