Usage

To run the program, we first must compile and link our program ww.c with other files, strbuf.h and strbuf.c which provides functionality regarding string manipulation such as concatenation.

To do this, we run make ww

Testing Strategy

We developed the solutions directory (which contains the solutions to the test cases) by manually adding the breaks after a width (1st argument) w and making sure there was no additional new-lines and spaces. This is due to our driver program, driver, checking character by character, thus it has to exactly match. The driver program is a driver based driver.c which compares all the generated wrap.* files with the correct solutions in solutions/sol.*. driver is used to test the case of when the 2nd argument is a directory.

To develop the solutions, we use notepad++ which gives us the number of characters in a line to make sure it does not exceed the width size. We also utilized fold command in Linux which gives us a rough estimate (not exact), on how our output should look which we used as reference. For the purposes of debugging, we added a dashed line with the size of the width as the first line for the sake of debugging and quickly evaluating whether our program is not printing/writing more than the width. That is how we determined the correctness of our program.

We also utilized cmp and diff to compare the output of our program. Furthermore, we created make recipes with cmp and diff for the sake of testing when the 2nd argument is a file from the testcases

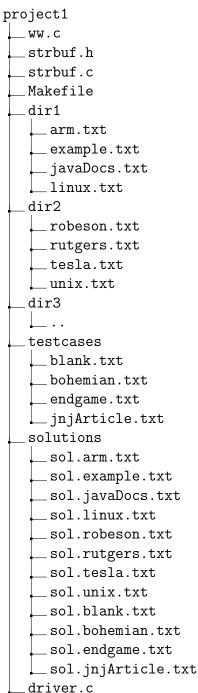
We also use the compiler flag -g -std=c99 -Wall -Wvla -fsanitize=address, undefined to catch memory leaks, possible errors, and debugging features. We also utilized valgrind to detect any memory leaks.

For error-handling invalid input and edge cases, we use perror with a corresponding ERRNO value from error. h to output a message in the case of an error. We also return EXIT_FAILURE which our Makefile and driver lets us know.

The type of files we ran with is with any text files such as .txt and for the edge case, .java. We also tried with files that do not have an extension. We also tried with directories. We played around with the permissions to see if our program would break if a file or folder did not have read/write permissions using chmod. Below are the scenarios and edge cases we ran.

File Structure for Testing

This is the file structure of our testing/debugging workspace.



Test Cases

The following are a subset of cases we ran with by creating a make recipe called test with a number appended at the end

• Testing File 1

- ./ww 49 testcases/endgame.txt | diff solutions/sol.endgame.txt -
- Testing File 2
 - ./ww 35 testcases/bohemian.txt | diff solutions/sol.bohemian.txt -
- Testing File 3
 - ./ww 24 testcases/jnjArticle.txt | diff solutions/sol.jnjArticle.txt -
- Testing dir1
 - ./ww 25 dir1
 - ./driver dir1
- Testing dir2
 - ./ww 30 dir2
 - ./driver dir2
- Negative Width
 - ./ww -24 testcases/endgame.txt
- Invalid Width
 - ./ww 20Hi testcases/endgame.txt
- Zero Width
 - ./ww 0 testcases/endgame.txt
- Width of 1
 - ./ww 1 testcases/endgame.txt
- Width of 2
 - ./ww 2 testcases/endgame.txt
- Inputting a very large width (such as 300 which is bigger than our buffer size)
 - ./ww 1000 testcases/endgame.txt
- Files with No Permission
 - We used touch and chmod 0 to create and modify the new file cat and remove all permissions to determine whether the program crashes if a file with no permissions has been passed.
 - ./ww 300 cat

- Directories with No Permission
 - We used touch and chmod 0 to create and modify a new directory called foo and remove all permissions to determine whether the program crashes if a directory with no permissions has been passed.
 - ./ww 300 foo
- Empty Directory
 - dir3 is an empty directory
 - ./ww 300 dir3
- Blank File
 - We passed a blank file to see what would it output in regards of spacings and newlines
- Directory within a directory with files
 - make test14
- Directory within a directory with files
 - This is to check whether our program does not recursive within another directory; we check that it ignores the directory
- Passing Relative and Absolute Paths
- Passing dir1, dir1/, /dir1 where "dir1" is a directory name
- Check for no duplicate "wrap." files and make sure it overwrites the previous "wrap." file
- Blank File with its cursor anywhere besides line 1
- A File With At Least A Character At The End With Many Spaces And New Lines Preceding
- A File With At Least A Character At The Beginning With Many Spaces And New Lines Succeeding
- A Blank File with its cursor anywhere besides line 1
- Directory With n Files And With x Files That Have No Permissions where x < n