Word Wrap

A brief description of your testing strategy. How did you determine that your program is correct? What sorts of files and scenarios did you check?

Testing strategy

General Properties of the Algorithm

1. For Word Wrap to `Wrap Correctly` the following properties must be satisfied.
    - The bytes read from an input file must be ordered correctly in the wrapped output file.
    - The number of characters in each line excluding the newline character of the wrapped output file must be less than or equal to the `max_width`.
    - Each word in a line must be seperated by a space. Words cannot be broken down to a sequence of characters from one line to another. If the word can't fit in the line then its moved to the next line.
    - Each line must end with a newline including the last line.
    - The lines are normalized meaning extra white spaces in the line is replaced by a single white space, and the line is trimmed so that there is no white space in the beginning or end of a sentence.
    - The breaks between lines are also normalized meaning extra breaks between consecutive lines is replaced with only one break.

How did we test our program?

1. Build the program by executing `make` from the root directory of this project.
2. To generate a wrapped file, run the `word_break` executable located in `bin` using the command `./bin/word_break <max-width> <test-file-path>`.
    - Note: `<test-file-path>` can either be a file, directory or nothing.
        - If it's a regular file then it will read the regular file, and write to `STDOUT`.
        - If it's a directory then it will traverse through the directory, read the file, and write to a new file with the prefix `wrap.<file-name>` in the directory.
        - If it's nothing then it will read input from `STDIN` and write the wrap text to `STDOUT`.
3. Now to check if the wrapped text is actually correct, we run the executable located in `bin` using the command `./bin/wcheck <max-width> <wrapped-output-file>`.
    - Note since the output is written to `STDOUT`, you can either redirect the output to a text file using the command `./bin/word_break <max-width> <test-file> > wrapped_<test_file>` and run the wcheck program using `./bin/wcheck <max_width> wrapped_<test_file>` or you can pipe (easier, in my opinion) `word_break` output into `wcheck` using the command `./bin/word_break <max-width> <test-file> | ./bin/wcheck <max-width>`.

Testing the algorithm

1. Empty file (`0 bytes`)
    - **Result**: The program return an empty file.
2. Paragraph with a single `break`
    - **Result**: The program will wrap
3. Paragraph with multiple `breaks`
    - multiple single breaks between sentence
        - **Result**: Wraps correctly, and single break remains a single break.
    - multiple multiple breaks between sentence.
        - **Result**: Wraps correctly, and multiple breaks become a single break
4. One word per line.
    - **Result**: Wraps correctly. Each word in the line ends with a newline character. If the word length exceeds the `max_width` then `EXIT_FAILURE` is returned internally by the process (look at point 4 in Testing the errors section).
5. Skipping files that start with `wrap.` or `.`
    - **Result**: Returns Nothing, but skips over the file when traversing the directory
6. Varying read buffer-size does not affect the algorithm.
    - **Result**: Wraps correctly.
7. Abnormal whitespace locations in random sentence of the paragraphs.
    - **Result**: Normalizes the sentence by removing any extra white spaces in the trailing, leading or any portion of the sentence.
8. Large `max_width` value must wrap all lines into `one` single line.
    - **Result**: Wraps correctly.
9. Rewrapping a file with the same `max_width` used to wrap that file with the same `max_width` before.
    - **Result**: Wraps correctly. Identical file is returned. If the `cmp <wrapped-file> <re-wrapped-file>` program returns nothing then `<wrapped-file>` and `<re-wrapped-file>` is a identical file.
10. Minimum non-empty file (only 1 byte).
    - **Result**: Wraps correctly. The output has two characters, the existing 1 byte and the terminating newline character.
11. A file with only non-alphanumeric characters (e.g. \t, \n, etc).
    - **Result**: Wraps correctly. The output is 0 bytes as there were no alphanumeric characters in the input at all.

Testing the errors

1. Invalid file or directory path
    - **Result**: Invalid File Path Error and returns `EXIT_FAILURE`
2. Reading files with no read permission
    - **Result**: Permission Denied Error Message and returns `EXIT_FAILURE`
3. Writing files with no write permission
    - **Result**: Permission Denied Error Message and returns `EXIT_FAILURE`
4. Words that exceed `max_width`

- **Result**: Program continues to wrap the file, however the process that's executing `word_break` will return an `EXIT_FAILURE`. This can be checked by executing the program `echo $?` to see whether the latest executed program returned an `EXIT_FAILURE` or `EXIT_SUCCESS`.

Terminology

1. a `break` is defined as two newlines between a consecutive sentence

Authors

Rohan Deshpande (ryd4), and Selin Denise Altiparmak (sda81).