COM2002 INTERMEDIATE PROGRAMMING
2024 – 2025 SPRING

Laboratory Week: 28 April – 02 May 2025

Topic **:** Writing Large Programs

Program : Justify

Definition : The program deletes extra spaces and blank lines as well as filling and justifying lines.

- "Filling" a line means adding words until one more word would cause the line to overflow.
- "Justifying" a line means adding extra spaces between words so that each line has exactly the same length (60 characters).
    - Justification must be done so that the space between words in a line is equal (or nearly equal).
    - The last line of the output won't be justified.

Assume that no word is longer than 20 characters, including any adjacent punctuation. If the program encounters a longer word, it must ignore all characters after the first 20, replacing them with a single asterisk(*).

- The program can't write words one by one as they're read.
- Instead, it will have to store them in a "line buffer" until there are enough to fill a line.

The program will be split into three source files:

- *word.c*: functions related to words
- *line.c*: functions related to the line buffer
- *justify.c*: contains the main function

The program will also need two header files:

- *word.h*: prototypes for the functions in word.c
- *line.h*: prototypes for the functions in line.c

The *word.h* will contain the prototype for a function that reads a word. Reads the next word from the input and stores it in *word*. Makes *word* empty if no word could be read because of end-of-file. Truncates the *word* if its length exceeds *len*. The word.h header file has a prototype for only one function, *read_word*. *read_word* is easier to write if we add a small "helper" function, *read_char*. *read_char*'s job is to read a single character and, if it's a new-line character or tab, convert it to a space.

The *line.h* header file will contain the prototype for functions that perform the following operations:

- Write contents of line buffer without justification (*flush_line*)
- Determine how many characters are left in line buffer (*space_remaining*)
- Write contents of line buffer with justification (*write_line*)
- Clear line buffer (*clear_line*)
- Add word to line buffer (*add_word*)

Sample input:

C is quirky, flawed, and an
enormous success. Although accidents of history
surely helped, it evidently satisfied a need

for a system implementation language efficient
enough to displace assembly language,
yet sufficiently abstract and fluent to describe
algorithms and interactions in a wide variety
of environments.

-- Dennis M. Ritchie

Expected output:

C is quirky, flawed, and an enormous success. Although accidents of history surely helped, it evidently satisfied a need for a system implementation language efficient enough to displace assembly language, yet sufficiently abstract and fluent to describe algorithms and interactions in a wide variety of environments. -- Dennis M. Ritchie

Modify the program that it reads from one text file and writes to another. Have the program obtain the names of both files from the command line.