# wordwrap

**COLLABORATORS**

| | | | |
|---|---|---|---|
| | *TITLE* :<br><br>wordwrap | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | | April 14, 2022 | |

**REVISION HISTORY**

| NUMBER | DATE | DESCRIPTION | NAME |
|---|---|---|---|
| | | | |

# Contents

# Chapter 1

# wordwrap

## 1.1 main

```
                                 WordWrap v2.3
                             =============
                         © 1996-99 Wilhelm Nöker


Introduction
~~~~~~~~~~~~
"WORDWRAP" is a text filter that rearranges the line breaks in a plain
ASCII text, preserving paragraphs (or merging them into one, if you wish
;-).  I wrote it, because I often encounter strangely-formatted, but
large texts on Usenet - and because even with an editor like CygnusEd,
TurboText or EdWord that can reformat paragraphs, reformatting a whole
document is still a rather tedious job.

Select a topic for more info:
                 How to use it

                 Examples

                 Known problems

                 Contacting the author

                 Thanks and credits
```

## 1.2 usage

```
                "WordWrap" is invoked from the CLI.  It processes its files from
standard input to standard output, so you have to use Shell file
redirection:

  wordwrap  [options]  <original_file  >new_file

Options are:
```

```
    -l<len>         output line length, defaults to 75
    -b              protect
            blank lines
                    -bc[<indent>]   convert blank lines to indentation
    -bC[<indent>]   like -bc, but don't indent after groups of blank lines
    -i              protect
            indentation
                    -i<indent>       " , enforcing a fixed indentation width
    -ia             \_ the same as -i, but adding a
    -ia<indent>     /  blank line before each paragraph.
    -ic             convert indentations to blank lines
    -t<tabsize>     how to expand tab characters in the left margin
    -m<width>       add a
            left margin
                    -M<width>       ignore a left margin on the input
    -s<maxlen>      protect
            short input lines
                    -e<parno>       enable
            escape mode
             for one paragraph
    -h              try to put
            hyphenated words
             back together
    -H              like -h, but never removes the hyphens themselves
    -w
            two spaces
             after ".", "!" and "?"
    -W              two spaces after ".", "!", "?" and ":"
    -n              make non-breakable space (alt-sp: 0xa0) breakable
    -a<dstring>
            delimiter string
             that will always start a new line
    -z<dstring>      " , always ending an output line
    -A<dword>       like -a, but only matches against whole words
    -Z<dword>       like -z,  "
```

As you can see, options are CASE SENSITIVE.

## 1.3 Blank Lines

The -b option that preserves blank lines is highly recommended in all cases.  You will find very few texts that look good without it.

The -bc and -bC options are intended to save vertical space when printing texts with many small paragraphs.  The start of a new paragraph will be indicated by an indented line, while blank lines between the paragraphs will be dropped.

Groups of more than one blank line will not disappear completely in this procedure, but they will shrink by one line.  With -bC, the line following such a group of blank lines will remain left aligned.  The reasoning is, that such paragraphs will still be seperated clearly enough by the remaining blank lines.  If you don't think so, use -bc, which will indent all paragraphs.

The default width for such newly generated indentations is 4 spaces.
You may override this setting by specifying e.g. -bc6 or -bC6.

## 1.4 indentation

```
  The -i option assumes that each indented line is the start of a new
paragraph, like in this section.  As a side effect, the line break of
indented or centered blocks of text will also be preserved by this
option.

        That
        may apply to
        addresses
                                 or
                              headings.
                              ~~~~~~~~~
   Tab characters encountered in this process are always converted to
spaces.  One tab produces 4 spaces by default, and you can override that
by specifying e.g.  -t8.  This conversion is necessary because otherwise
"WordWrap" couldn't determine the exact size of its output lines.
    Maybe you don't like the way each of these paragraphs is indented by
a different amount of spaces.  You could help that by specifying a fixed
indentation width, -i4 for example, which would indent anything in your
document that isn't left aligned by exactly 4 spaces.
     The -ia and -ic options are for people who feel that indentation is
an inappropriate or insufficient way of separating paragraphs, and who
prefer blank lines instead.
```

## 1.5 Left Margin

Please think twice before using the -m option. IMHO, adding a left margin to ASCII texts is a waste of space. However, it can be useful for printing, if you're too lazy to set a left margin in the printer prefs.

The -M<width> option is only needed (if ever) together with the -i options. It strips <width> blanks from the start of each input line before looking for indented lines. Note that tab signs count as multiple spaces, 4 by default.

B.t.w., the -M operation is almost the same as cutting a vertical block off the left margin from within a text editor, but not exactly the same: It is safer, because it will never remove any non-blank characters.

## 1.6 Short Input Lines

When you reformat texts with headings in them you may notice that dashes used for underlining are wrapped in an unwanted way. Try -s20, this will insert a line break after each input line containing less than 20 non-blank characters: voilà.

B.t.w., it's hard to do any damage with this option, so you might try -s40, too, or even -s60. Also note that the really wide headings usually take care of themselves, anyway, because they won't fit onto the same line as their underlining dashes at all.

## 1.7 Escape Mode

The -i and -s options cannot fully prevent that the alignment of tables or ASCII art diagrams is recklessly destroyed. You might either restore such sections manually or insert a copy from the original document. But it's easier to keep "WordWrap" from touching some paragraphs at all.

Paragraphs are specified by number for this purpose, where numbers start from zero and one or more blank lines mark the start of a new paragraph. (This method of counting is always used, no matter what -b or -i options are supplied.) So for example the text in this section could be reformatted using -e2 to keep the following table properly aligned.

```
  system    | advantage   | drawback
  ----------+-------------+------------------------------
  Amiga     | nice        | not always on sale
  Macintosh | reliable    | only one mouse button
  PC        | compatible  | requires frequent CPU updates
```

Finding out the correct number for a certain paragraph may require some trial and error, especially when longer texts are involved. It is highly recommended to perform such experiments without output file redirection at first: Escaped sections will then appear highlighted in the console output to help you see what's going on.

## 1.8 Undo Hyphenation

Reformatting texts with hyphen- ations in them may look rather crappy. Now don't say that plain ASCII texts aren't supposed to contain hyphenated words anyway. Teletext pages saved as ASCII do, for example, as well as some of the Project Gutenberg etexts that I've seen. (I suppose that's because they were scanned from printed copies that were typeset that way.)

The -h option will try to put such hyphenated words back together. This is a little tricky, because not all dashes in a text are meant as hyphens, and not all hyphens were introduced only for splitting a word across lines. Wordwrap will stick to the following rules, hoping to do the right thing:

- a hyphen is always the last non-blank character on an input line
- the character preceding it and the first non-blank character on the next line must be a digit or a letter
- unless both the preceding and the following character are lowercase letters, the concerning words are joined, but the hyphen is not

   removed

In some texts only such words are split across lines which contained a dash anyway, like "thirty-five", "half-frozen" or "high-strung" (in the Project Gutenberg version of Jack London's "Call of the Wild", for example). But as you may have already guessed, WordWrap would still remove the hyphens when putting these words back together, and consequently always be wrong in doing so. That's where the -H option comes in (always keep the hyphen), but I don't think you'll need it very often.

## 1.9 Space Between Sentences

It is common practice in English texts to separate sentences by an extra wide space. With -w enabled, "WordWrap" will try its best to find these places, by following some simple rules: Sentences are ended by '.', '!' or '?' at the end of a word, but never by a single letter or by a word consisting of two consonants only. This works surprisingly well, even with most abbreviations.

## 1.10 Delimiter Strings

To see how the "delimiter strings" work, you might try reformatting some C source with -a/* -z*/ -a{ -z} -a# "-z;" or -Aelse -Zelse. (The double quotes around "-z;" are needed, because otherwise your shell would treat the semicolon and the rest of the input line as a comment!)

I don't know if there are many useful applications to this feature, but I needed it to reformat fortune cookie files (using -a%% -z%%).

At the moment, you may supply up to 20 words of each category, which I think is plenty. Let me know if I'm wrong.

## 1.11 examples

You should have received four sample text files with this program (three of them lifted from Project Gutenberg etexts, aminet/docs/etext/, one from the web pages at Louisiana State University). Note that most of the following examples do not redirect their output, because I think playing with the options is nicer this way.

The first file, 'Treasure.txt', is one of my favourite chapters from "Treasure Island". It is in a very well-behaved format already, with blank lines seperating the paragraphs.

```
  wordwrap <treasure.txt -b -l40         ; wow, it really works :)
  wordwrap <treasure.txt -b -l60 -i      ; preserves the centered heading
  wordwrap <treasure.txt -bc             ; very compact output format
  wordwrap <treasure.txt -bC             ; see the difference?
  wordwrap <treasure.txt -bC -l65 >PRT:  ; print it, if you like
```

'SunTzu.txt' is an episode from the life of Sun Tzu, author of "The Art of War", with an interesting point to it (or at least some black humor). Most paragraphs are seperated by indentation only, and because it was originally a quotation inside a larger text, the whole thing is accompanied by a left margin of 2 spaces. Not so simple a format, but still one that "wordwrap" can be made understand.

```
  wordwrap <suntzu.txt -b                  ; nice try :->
  wordwrap <suntzu.txt -b -s30             ; for the underlined heading
  wordwrap <suntzu.txt -b -s30 -i          ; looks good?
  wordwrap <suntzu.txt -b -s30 -i -l40     ; no, something's wrong
  wordwrap <suntzu.txt -b -s30 -i -l40 -M2 ; this does the trick!
  wordwrap <suntzu.txt -b -s30 -ic -M2     ; like it better this way?
```

Then there's "HGWells.txt", a chapter from the classic "War of the Worlds". It contains hyphenations, so let's see what -h can do.

```
  wordwrap <hgwells.txt -b        ; "van- ished", "mul- titude": ouch
  wordwrap <hgwells.txt -b -h     ; see? it works
  wordwrap <hgwells.txt -b -h -w  ; add wide spaces between sentences
```

The last example 'CostOfWar.txt' was originally in HTML format and has already undergone some heavy editing. Reformatting it doesn't make much sense either, because its large tables leave you little choice about the line length. But at least it is a good example for wordwrap's escape mode.

```
  wordwrap <costofwar.txt -b -s50       ; close, but not perfect :->
  wordwrap <costofwar.txt -b -s50 -e3 -e6 -e7 -e13  ; save the tables
```

## 1.12 Known Problems

- o Delimiter words (-aAzZ) ending with a dash may collide with the operation of the -h/-H option. I don't think this will ever occur, but I just felt I should mention it.
- o GUI frontend: Yes, I know, it's badly needed, but I just don't have the time.

## 1.13 Contacting the Author

"WordWrap" is Freeware/Mailware: If you like it, send me e-mail (or a postcard):

```
        Wilhelm Nöker                    Hertastr. 8
        wnoeker@t-online.de              44388 Dortmund
                                         Germany
```

Postcards are nicer for me, of course, whereas e-mail is more likely to get a reply.

## 1.14 Thanks and Credits

```
"WordWrap" was written using CygnusEd 4.2 and the GNU C compiler.

Thanks to K.E. Trygstad, Adam M. Costello, Jim Aites and Kristoffer
Karlsson for helpful comments and suggestions.

Greetings to Jen Allen, wherever you are.
```