

Line wrap and word wrap

From Wikipedia, the free encyclopedia

Line breaking, also known as **word wrapping**, is the process of breaking a section of text into lines such that it will fit in the available width of a page, window or other display area. In text display, **line wrap** is the feature of continuing on a new line when a line is full, such that each line fits in the viewable window, allowing text to be read from top to bottom without any horizontal scrolling. **Word wrap** is the additional feature of most text editors, word processors, and web browsers, of breaking lines between words rather than within words, when possible. Word wrap makes it unnecessary to hard-code newline delimiters within paragraphs, and allows the display of text to adapt flexibly and dynamically to displays of varying sizes.


Contents

- 1 Soft and hard returns
 - 1.1 Unicode
- 2 Word boundaries, hyphenation, and hard spaces
- 3 Word wrapping in text containing Chinese, Japanese, and Korean
- 4 Algorithm
 - 4.1 Minimum number of lines
 - 4.2 Minimum raggedness
 - 4.3 History
- 5 See also
- 6 References
- 7 External links
 - 7.1 Knuth's algorithm
 - 7.2 Other word-wrap links

Soft and hard returns

A soft return or soft wrap is the break resulting from line wrap or word wrap (whether automatic or manual), whereas a hard return or hard wrap is an intentional break, creating a new paragraph. With a hard return, paragraph-break formatting can (and should) be applied (either indenting or vertical whitespace). Soft wrapping allows line lengths to adjust automatically with adjustments to the width of the user's window or margin settings, and is a standard feature of all modern text editors, word processors, and email clients. Manual soft breaks are unnecessary when word wrap is done automatically, so hitting the "Enter" key usually produces a hard return.

Alternatively, "soft return" can mean an intentional, stored line break that is not a paragraph break. For example, it is common to print postal addresses in a multiple-line format, but the several lines are understood to be a single paragraph. Line breaks are needed to divide the words of the address into lines of the appropriate length.

In the contemporary graphical word processors Microsoft Word and OpenOffice.org, users are expected to type a carriage return ( Enter) between each paragraph. Formatting settings, such as first-line indentation or spacing between paragraphs, take effect where the

carriage return marks the break. A non-paragraph line break, which is a soft return, is inserted using ⇧ Shift + ↵ Enter or via the menus, and is provided for cases when the text should start on a new line but none of the other side effects of starting a new paragraph are desired.

In text-oriented markup languages, a soft return is typically offered as a markup tag. For example, in HTML there is a `
` tag that has the same purpose as the soft return in word processors described above; the `<p>` tag is used to contain paragraphs.

Unicode

The Unicode Line Breaking Algorithm determines a set of positions, known as *break opportunities*, that are appropriate places in which to begin a new line. The actual line break positions are picked from among the break opportunities by the higher level software that calls the algorithm, not by the algorithm itself, because only the higher level software knows about the width of the display the text is displayed on and the width of the glyphs that make up the displayed text.^[1]

The Unicode character set provides a line separator character as well as a paragraph separator to represent the semantics of the soft return and hard return.

0x2028 LINE SEPARATOR

* may be used to represent this semantic unambiguously

0x2029 PARAGRAPH SEPARATOR

* may be used to represent this semantic unambiguously

Word boundaries, hyphenation, and hard spaces

The soft returns are usually placed after the ends of complete words, or after the punctuation that follows complete words. However, word wrap may also occur following a hyphen inside of a word. This is sometimes not desired, and can be blocked by using a non-breaking hyphen, or hard hyphen, instead of a regular hyphen.

A word without hyphens can be made wrappable by having soft hyphens in it. When the word isn't wrapped (i.e., isn't broken across lines), the soft hyphen isn't visible. But if the word is wrapped across lines, this is done at the soft hyphen, at which point it is shown as a visible hyphen on the top line where the word is broken. (In the rare case of a word that is meant to be wrappable by breaking it across lines but *without* making a hyphen ever appear, a zero-width space is put at the permitted breaking point(s) in the word.)

Sometimes word wrap is undesirable between adjacent words. In such cases, word wrap can usually be blocked by using a hard space or non-breaking space between the words, instead of regular spaces.

Word wrapping in text containing Chinese, Japanese, and Korean

In Chinese, Japanese, and Korean, word wrapping can usually occur before and after any Han character, but certain punctuation characters are not allowed to begin a new line.^[2] Japanese kana, letters of the Japanese alphabet, are treated the same way as Han Characters (Kanji) by extension, meaning words can, and tend to be broken without any hyphen or other indication that this has happened.

Under certain circumstances, however, word wrapping is not desired. For instance,

- word wrapping might not be desired within personal names, and
- word wrapping might not be desired within any compound words (when the text is flush left but only in some styles).

Most existing word processors and typesetting software cannot handle either of the above scenarios.

CJK punctuation may or may not follow rules similar to the above-mentioned special circumstances. It is up to line breaking rules in CJK.

A special case of line breaking rules in CJK, however, always applies: line wrap must never occur inside the CJK dash and ellipsis. Even though each of these punctuation marks must be represented by two characters due to a limitation of all existing character encodings, each of these are intrinsically a single punctuation mark that is two ems wide, not two one-em-wide punctuation marks.

Algorithm

Word wrapping is an optimization problem. Depending on what needs to be optimized for, different algorithms are used.

Minimum number of lines

A simple way to do word wrapping is to use a greedy algorithm that puts as many words on a line as possible, then moving on to the next line to do the same until there are no more words left to place. This method is used by many modern word processors, such as OpenOffice.org Writer and Microsoft Word. This algorithm always uses the minimum possible number of lines but may lead to lines of widely varying lengths. The following pseudocode implements this algorithm:

```
SpaceLeft := LineWidth
for each Word in Text
  if (Width(Word) + SpaceWidth) > SpaceLeft
    insert line break before Word in Text
    SpaceLeft := LineWidth - Width(Word)
  else
    SpaceLeft := SpaceLeft - (Width(Word) + SpaceWidth)
```

Where `LineWidth` is the width of a line, `SpaceLeft` is the remaining width of space on the line to fill, `SpaceWidth` is the width of a single space character, `Text` is the input text to iterate over and `Word` is a word in this text.

Minimum raggedness

A different algorithm, used in TeX, minimizes the sum of the squares of the lengths of the spaces at the end of lines to produce a more aesthetically pleasing result. The following example compares this method with the greedy algorithm, which does not always minimize squared space.

For the input text

```
aaa bb cc ddddd
```

with line width 6, the greedy algorithm would produce:

```
----- Line width: 6
aaa bb   Remaining space: 0
cc       Remaining space: 4
ddddd    Remaining space: 1
```

The sum of squared space left over by this method is $0^2 + 4^2 + 1^2 = 17$. However, the optimal solution achieves the smaller sum $3^2 + 1^2 + 1^2 = 11$:

```
----- Line width: 6
aaa      Remaining space: 3
bb cc    Remaining space: 1
ddddd    Remaining space: 1
```

The difference here is that the first line is broken before `bb` instead of after it, yielding a better right margin and a lower cost 11.

By using a dynamic programming algorithm to choose the positions at which to break the line, instead of choosing breaks greedily, the solution with minimum raggedness may be found in time $O(n^2)$, where n is the number of words in the input text. Typically, the cost function for this technique should be modified so that it does not count the space left on the final line of a paragraph; this modification allows a paragraph to end in the middle of a line without penalty. It is also possible to apply the same dynamic programming technique to minimize more complex cost functions that combine other factors such as the number of lines or costs for hyphenating long words.^[3] Faster but more complicated linear time algorithms based on the SMAWK algorithm are also known for the minimum raggedness problem, and for some other cost functions that have similar properties.^{[4][5]}

History

A primitive line-breaking feature was used in 1955 in a "page printer control unit" developed by Western Union. This system used relays rather than programmable digital computers, and therefore needed a simple algorithm that could be implemented without data buffers. In the Western Union system, each line was broken at the first space character to appear after the 58th character, or at the 70th character if no space character was found.^[6]

The greedy algorithm for line-breaking predates the dynamic programming method outlined by Donald Knuth in an unpublished 1977 memo describing his TeX typesetting system^[7] and later published in more detail by Knuth & Plass (1981).

See also

- Word divider
- Non-breaking space
- Zero-width space

References

1. Heninger, Andy, ed. (2013-01-25). "Unicode Line Breaking Algorithm" (<http://www.unicode.org/L2/L2013/13022-uax14-31.pdf>) (PDF). *Technical Reports*. Annex #14 (Proposed Update Unicode Standard); 2. Retrieved 10 March 2015. "WORD JOINER should be used if the intent is to merely prevent a line break"
2. Lunde, Ken (1999), *CJKV Information Processing: Chinese, Japanese, Korean & Vietnamese Computing* (<https://books.google.com/books?id=Cn7jnk9WwZEC&pg=PA352>), O'Reilly Media, Inc., p. 352, ISBN 9781565922242.
3. Knuth, Donald E.; Plass, Michael F. (1981), "Breaking paragraphs into lines", *Software: Practice and Experience*, 11 (11): 1119 – 1184, doi:10.1002/spe.4380111102 (<https://doi.org/10.1002%2Fspe.4380111102>).
4. Wilber, Robert (1988), "The concave least-weight subsequence problem revisited", *Journal of Algorithms*, 9 (3): 418 – 425, MR 955150 (<https://www.ams.org/mathscinet-getitem?mr=955150>), doi:10.1016/0196-6774(88)90032-6 (<https://doi.org/10.1016%2F0196-6774%2888%2990032-6>).
5. Galil, Zvi; Park, Kunsoo (1990), "A linear-time algorithm for concave one-dimensional dynamic programming", *Information Processing Letters*, 33 (6): 309 – 311, MR 1045521 (<https://www.ams.org/mathscinet-getitem?mr=1045521>), doi:10.1016/0020-0190(90)90215-J (<https://doi.org/10.1016%2F0020-0190%2890%2990215-J>).
6. Harris, Robert W. (January 1956), "Keyboard standardization" (<http://massis.lcs.mit.edu/archives/technical/western-union-tech-review/10-1/p040.htm>), *Western Union Technical Review*, 10 (1): 37 – 42.
7. Knuth, Donald (1977), *TEXDR.AFT* (<http://www.saildart.org/TEXDR.AFT%5B1,DEK%5D>), retrieved 2013-04-07. Reprinted in Knuth, Donald (1999), *Digital Typography*, CSLI Lecture Notes, 78, Stanford, California: Center for the Study of Language and Information, ISBN 1-57586-010-4.

External links

- Unicode Line Breaking Algorithm (<http://www.unicode.org/reports/tr14/>)

Knuth's algorithm

- "Knuth & Plass line-breaking Revisited" (<http://defoe.sourceforge.net/folio/knuth-plass.html>)
- "tex_wrap": "Implements TeX's algorithm for breaking paragraphs into lines." (<http://oedipus.sourceforge.net/texlib/>) Reference: "Breaking Paragraphs into Lines", D.E. Knuth and M.F. Plass, chapter 3 of *_Digital Typography_*, CSLI Lecture Notes #78.
- Text::Reflow - Perl module for reflowing text files using Knuth's paragraphing algorithm. (<https://metacpan.org/module/Text::Reflow>) "The reflow algorithm tries to keep the lines the same length but also tries to break at punctuation, and avoid breaking within a proper name or after certain connectives ("a", "the", etc.). The result is a file with a more "ragged" right margin than is produced by fmt or Text::Wrap but it is easier to read since fewer phrases are broken across line breaks."
- adjusting the Knuth algorithm (<https://web.archive.org/web/20070930015603/http://www.nabble.com/Initial-soft-hyphen-support-t2970713.html>) to recognize the "soft hyphen".
- Knuth's breaking algorithm. (<http://wiki.apache.org/xmlgraphics-fop/KnuthsModel>) "The detailed description of the model and the algorithm can be found on the paper "Breaking Paragraphs into Lines" by Donald E. Knuth, published in the book "Digital Typography" (Stanford, California: Center for the Study of Language and Information, 1999), (CSLI Lecture Notes, no. 78.)" ; part of Google Summer Of Code 2006 (<http://wiki>.

apache.org/xmlgraphics-fop/GoogleSummerOfCode2006/FloatsImplementationProgress)

- "Bridging the Algorithm Gap: A Linear-time Functional Program for Paragraph Formatting" (<http://citeseer.ist.psu.edu/23630.html>) by Oege de Moor, Jeremy Gibbons, 1999

Other word-wrap links

- the reverse problem -- picking columns just wide enough to fit (wrapped) text (<http://www.codecomments.com/message230162.html>) (Archived version (<https://archive.is/20070927021648/http://www.codecomments.com/message230162.html>))
- KWordWrap Class Reference (<http://api.kde.org/4.x-api/kdelibs-apidocs/kdeui/html/classKWordWrap.html>) used in the KDE GUI
- "Knuth linebreaking elements for Formatting Objects" (<http://www.leverkruid.eu/GKPLinebreaking/elements.html>) by Simon Pepping 2006. Extends the Knuth model to handle a few enhancements.
- "Page breaking strategies" (<http://wiki.apache.org/xmlgraphics-fop/PageLayout/>) Extends the Knuth model to handle a few enhancements.
- "a Knuth-Plass-like linebreaking algorithm (<http://www.techwhirl.com/archives/0504/techwhirl-0504-00203.html>) ... The *really* interesting thing is how Adobe's algorithm differs from the Knuth-Plass algorithm. It must differ, since Adobe has managed to patent its algorithm (6,510,441)."[1] (<http://www.techwhirl.com/archives/0504/techwhirl-0504-00206.html>)
- "Murray Sargent: Math in Office" (<http://blogs.msdn.com/murrays/archive/2006/11/15/lineservices.aspx>)
- "Line breaking" (<http://xyxyz.org/line-breaking/>) compares the algorithms of various time complexities.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Line_wrap_and_word_wrap&oldid=782097377"

Categories: Text editor features | Typography | Dynamic programming | Unicode algorithms

-
- This page was last edited on 2017-05-25, at 06:22:49.
 - Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.